1+1+n+1 to rule them all in the cloud!

Visual Studio ALM Rangers

Are you looking for a magic cure-all for planning team environments in Visual Studio?

If yes, **you are not alone**!

The Visual Studio ALM Rangers[1] have been dogfooding different models, infrastructures, and technologies since 2006 in search of a solution that meets their unique requirements for disconnected, distributed, and part-time project teams. The details are chronicled in ALM Guidance: Visual Studio ALM Rangers — Reflections on Virtual Teams[2]. While we cannot give you magic cure-all, we can share the process we followed to arrive at the team environment we enjoy in the "cloud" today.

Do you believe in dogfooding?

We do, and it has given us and the product teams great return on investments!

The ALM Rangers, as the name implies, are passionate about **ALM** tooling. We're veteran users of the tooling, such as Visual Studio and related technologies and services. We evangelise and guide the community so it's only logical that we use the same ALM tooling and, more importantly, that we evaluate new products before they become mainstream.

To summarise, the core advantages of dogfooding are:

- Access to the latest and greatest technologies.
- Active collaboration with the product teams and exchange of ideas, issues, and feedback from the field.
- Ability to influence the technology before it is released and prepare guidance for practical usage.

For the remainder of the article, we will step through the planning of our Team Project Collection, Team Project(s), Teams, and the one backlog to rule all teams.

Let the journey begin by reviewing the Team Foundation Server Planning Guide[3] which delivers practical and scenario-based guidance for the implementation of Team Foundation Server. It guides us through the decisions about whether to have one or more Team Foundation Servers, one or more Team Project Collections, one or more Team Projects, and one or more Teams, based on scenarios and implications of each decision.

v1.0 – 2013-04-19 aka.ms/vsarcloudhome

Contents

Willy-Peter Schaub is a Senior Program Manager with the Visual Studio ALM Rangers at the Microsoft Canada Development Center. Since the mid-'80s, he's been striving for simplicity and maintainability in software engineering. His blog is at *blogs.msdn.com/b/willy-peter_schaub* and twitter *www.twitter.com\wpschaub*.

On-premises or cloud-based home?

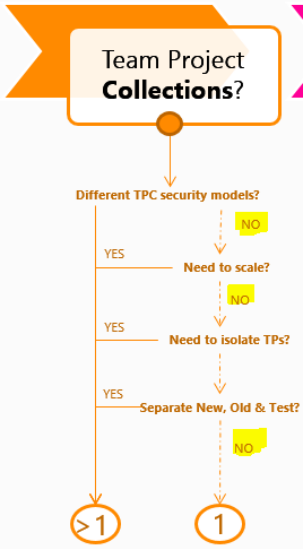
Although the planning guide and the planning poster would typically nudge us towards the cloud-based Team Foundation Service home, we had a different motivation. We were the first users of the Team Foundation Service[4], in the days when it was still known as tfspreview, and we committed ourselves to the solution. In fact, in April 2011, we moved all teams from on-premises environments to the service, with no easy or painless return strategy … we were that committed to making it work!

This document reflects two (2) years of dogfooding and adaptation experience. Hopefully there are a few nuggets or gems in here that you will find useful.

While we evaluated and tested the use of the TFS Integration Tools to migrate all our projects from on-premises servers to the cloud (see Migrating from an On-Premises Team Foundation Server to Team Foundation Service Preview Using the TFS Integration Tools[5] for details) we made two strategic choices. We decided to start at ground zero in terms of Work Items and we decided to manually check in the "latest" code base as needed. We have no auditing or operational requirements that required us to maintain extensive history and for the very infrequent visits to the past, our backups have proven more than adequate.

ALM Rangers dogfood and depend on the Team Foundation Service in the cloud.

One Team Project Collection (TPC) to organize them all



It is recommended to keep the number of Team Project Collections as small as possible to minimise administration cost and, most importantly, to keep it simple for the user. Switching between team projects collections closes all queries, work items, documents and solutions, so frequent switching can be a daunting, confusing, and unproductive experience.

For the Rangers, the decision to embrace the Team Foundation Service enforced the one team project collection model, because each service account is currently limited to one Team Project Collection. This implies that we are enjoying the advantages of administrative and usage simplicity, as well as the ability to share artefacts among Team Projects.

In the future, we might analyse the "potential" need for more Team Project Collections if the service introduces support for more than one per account. What we would like to change, when possible, is the Team Project Collection name to make it more descriptive.

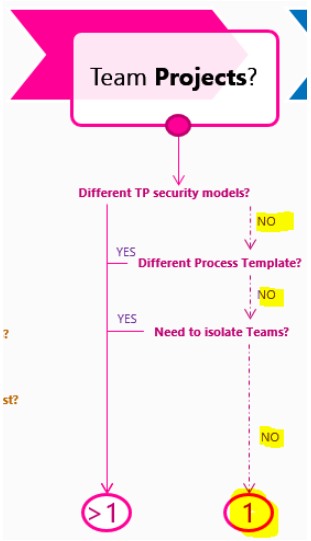
ALM Rangers live and thrive within one Team Project Collection.

One Team Project to contain them all

Planning of Team Projects is crucial, because once your design strategy has been implemented it becomes immutable and very, very difficult and costly to change. As outlined in ALM Rangers switching to single TPC/TP and multiple Teams model[6] our objectives were to improve the user experience, centralize our backlog and bug management, standardize on one process template, and most importantly, to dogfood and align our environment with our latest planning guidance.

Looking back over the past year, the move to a single Team Project has been interesting, sometimes challenging, but most definitely a good one. For the ALM Rangers who have fully embraced the environment and the supporting Rangers Ruck[7]

process, the productivity gain through less context sharing and better sharing, as well as the improved transparency across projects is both evident and appreciated.



In reality, our Team Project Collection currently contains nine (9) Team Projects. We started with a one Team Project per project model, are now in one Team Project per product (Visual Studio ALM), and anticipate that the collection will have no more than three (3) Team Projects in the future.

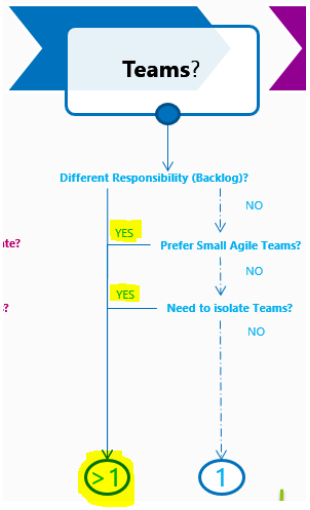
***Are we** really only **using one Team Project? Yes** and **no** …*

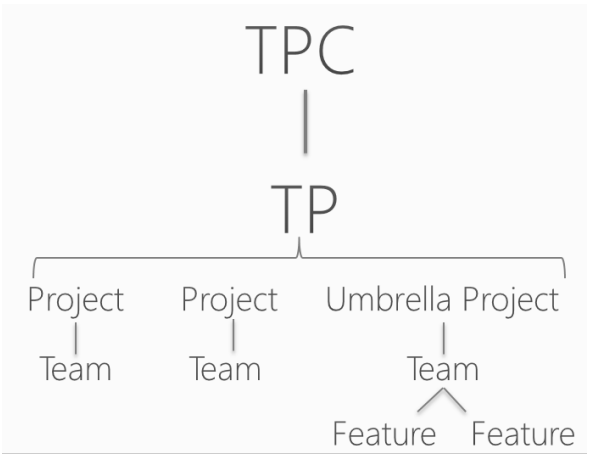
As shown in the illustration, we have categorised our Team Projects into Production, Archive, and Supporting roles. We only have one production Team Project, which means that 99% of the Rangers live in one Team Project Collection and one Team Project for all their ALM Ranger adventures. The Archive Team Projects are used infrequently as "read-only" reference excursions and the Supporting Team Projects will soon be collapsed to only one free-for-all experimentation sandbox. The Test Team Projects were used for specialised testing, which has a better home in temporary virtualised test environments. The Help Team Project started with a noble intent, but as it is hardly (if at all) used for Ruck Process training it too will be phased out to reduce noise, resource usage, and complexity.

ALM Rangers have one operational Team Project and a "few" archive and research Team Projects.

Multiple Teams lighting up

The introduction of Team Foundation Server Teams allows you to design an environment that is more representative of and conducive to a team environment. Working though the Team Foundation Server Planning Guide, we quickly realised that we needed more than one team, each owning an area path and iteration path. As part of our dogfooding we are encouraging teams to remain relatively small, typically five and up to a maximum of ten members, to promote agility and reduce administrative overhead on the project lead and Ruck Masters. Teams that outgrow these limits usually become tough to monitor within our challenging environment (ALM Guidance: Visual Studio ALM Rangers — Reflections on Virtual Teams) and are best broken into smaller feature teams.



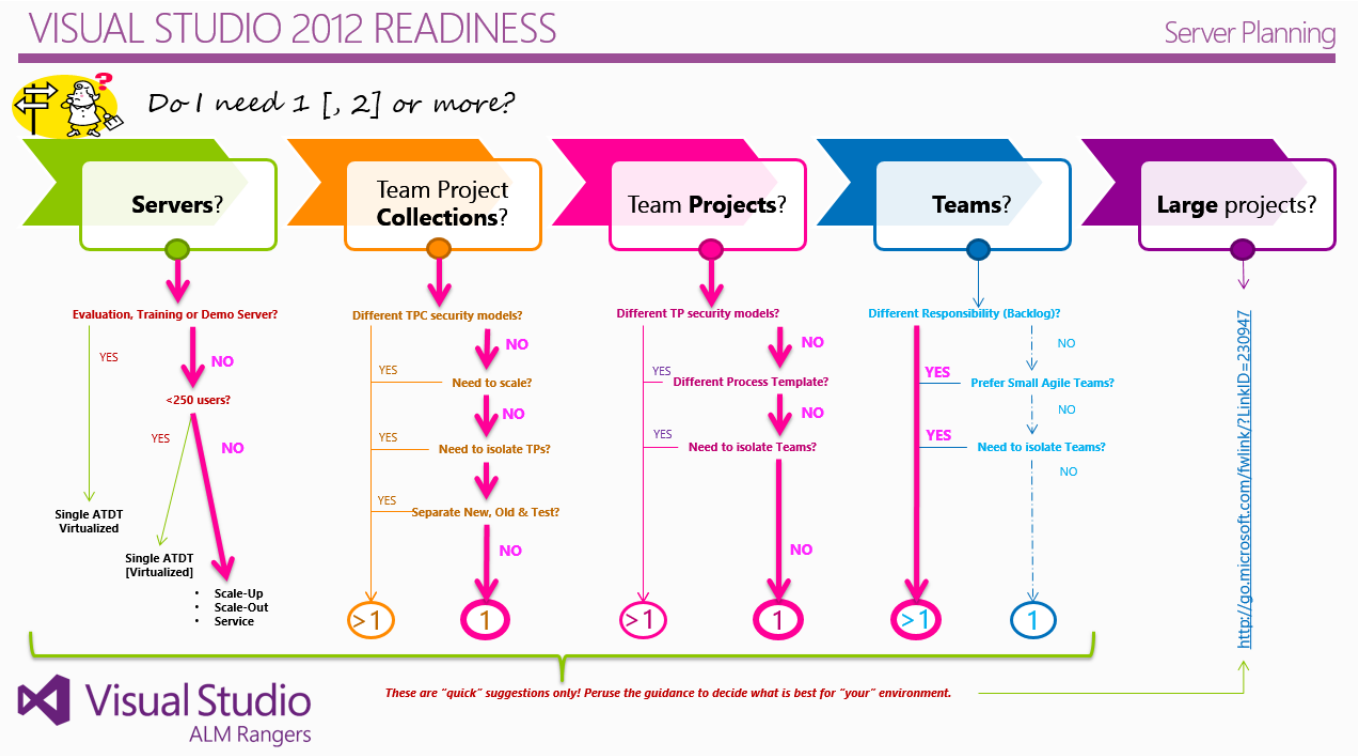


As shown, we have sliced up our Team Project into projects, typically relating 1:1 to those seen on ALM Ranger Solutions[8]. Each project is represented by one Team Foundation Server Team. In some cases, for example with our Quick Response Solutions[9], we have an umbrella project that contains multiple related but separate feature projects. We use the Area Path as the dividing line between feature teams, but treat them as one project, with one joint goal, tracking, and backlog.

ALM Rangers have many Teams and in some cases Feature Teams.

Planning at a glance

The following image consolidates our journey through the server planning quick reference poster, using the Team Foundation Server Planning Guide and sprinkle of "real world" constraints to decide how many servers, Team Project Collections, Team Projects and Teams we should be using.



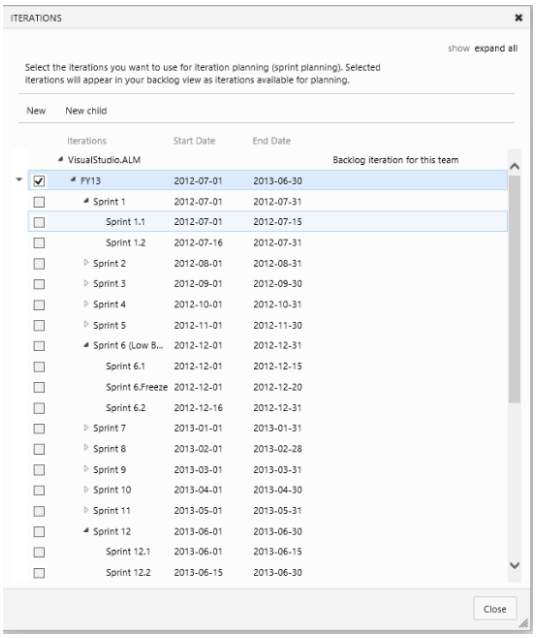
One Backlog to keep them all focused

This is where the ALM Rangers environment has really **lit up** in terms of planning, administration, and overall transparency … but we will explore the advantages, gems, and evidence in the next section. First let's understand how we have implemented **one backlog** to rule all our teams and feature teams.

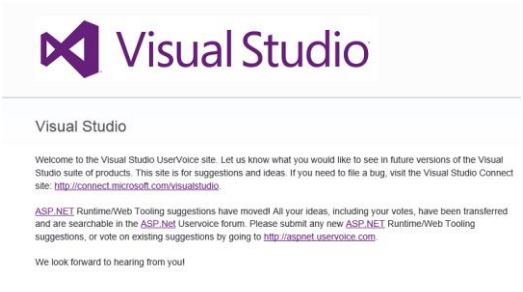
A backlog is a list of backlogged **work items**, which are assigned to an **iteration path** and an **area path**. In essence, we use the backlog to define **what** will be done **when** and by **whom**.

Shared Iteration Paths, but different cadence as chosen by the team … huh?

You can see that the Rangers break up the year into 12 monthly sprints, each of which is optionally split into two one-half month sprints. We started with the new definition of the iteration path on July 1, 2012, which is the reason why the monthly sprints are numbered 1-12 for monthly and 1.1, 1.2, 2.1, 2.2, etc., during FY13 and 13-24 in FY14. The teams either use monthly, half monthly, or a combination thereof. This allows each team to choose their delivery cadence. The typical pattern is a half month planning sprint, followed by 3 to 4 monthly sprints. Each sprint starts with achievable sprint objectives and ends with a sprint deliverable. Teams with no deliverable or heart-beat, instill no confidence and are red flagged.

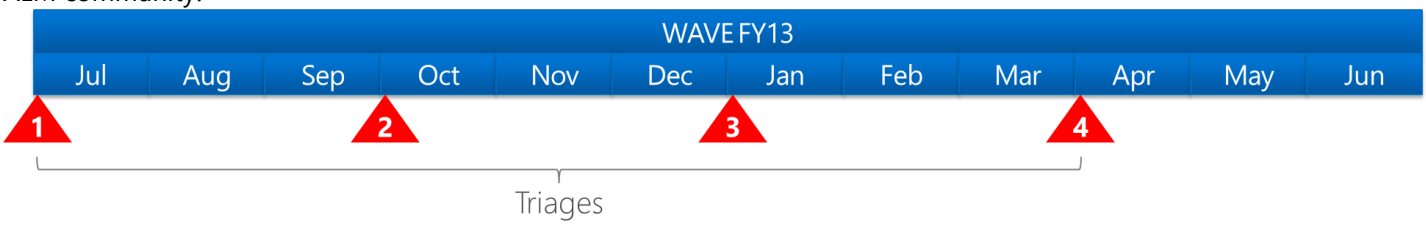


How we harvest and triage ideas

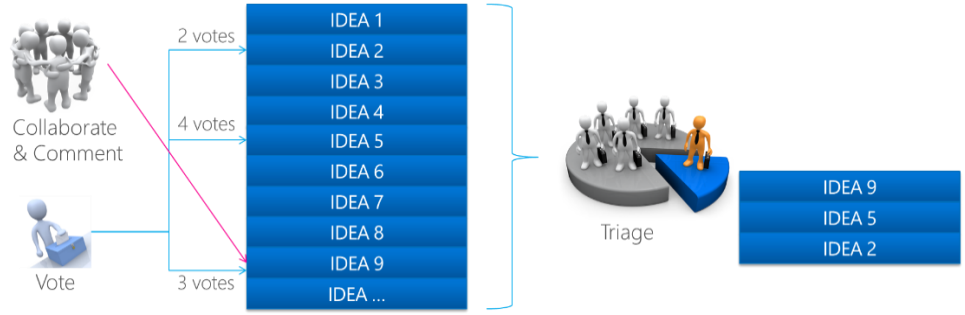


The ALM Rangers solution factory is fueled by the community. Through UserVoice[10] our users are a valuable source of innovative ideas.

We perform an "ideas triage" four times a year in January, April, July, and October. During each triage, we look for ideas with a lot of community support (votes), which are great candidates for out-of-band solutions for feature gaps and value-added guidance for the ALM community.



Each triage results in a couple of handfuls of ideas, which are triaged against strategic project ideas from the Product Group and the ALM Rangers.



The ideas that make it through the triage are then paired with product owners and project leads, who define and prioritise the requirements, which are defined as Epics on the backlog. Next follows a kick-off during which the product owner and project lead "sell" the project idea and associated Epics, looking for ALM Rangers to join their adventure (team).

Projects need full commitment by all stakeholders. Ideas that attract no product owner, project lead or team members are returned to the bucket for reconsideration at the next triage.

Mixing it all together into a single backlog

The posts How the ALM Rangers are using Visual Studio ALM … in a practical nutshell[11] and ALM Rangers ALM Dogfooding and the "Angst" Factor – Part 3[12] walk through the dogfooding environment, the single Team Project, and backlog strategy in greater detail. In this overview, we will merely highlight that the resultant backlog has a queue of Epics that are prioritised and then assigned to teams and iteration paths. This backlog defines the scope and the timeframe in which the solutions will be created and delivered.

Dealing with uninvited guests … bugs!

Do you also experience uninvited guests and anomalies, more commonly known as "bugs"? Rest assured, we endure them on a regular basis … in fact we frown upon projects that are not infested by bugs, because rigorous, disciplined, and focused testing typically uncovers lots of the critters.

These are the steps we perform with bugs:

- **Prefix** the title of bugs with "**Bug:**" so that they are evident on the backlog and especially the task board.
- Ensure the **title** is **self-explanatory** and that the description and attachments add adequate context.
- Set **Area Path** to the appropriate team or feature team.
- **Assign** the bug **to** the **project lead**, who will triage and re-assign to the appropriate team member.
- During the bug triage the bug is **prioritised** and assigned to the relevant **iteration**.
- In addition, the team can define one or more **child tasks** if the bug needs to be broken down into discrete tasks.

To read more about our unique way of tracking and dealing with bugs, peruse ALM Rangers ALM Dogfooding … dealing with bugs on tfs.visualstudio.com – Part 8[13].

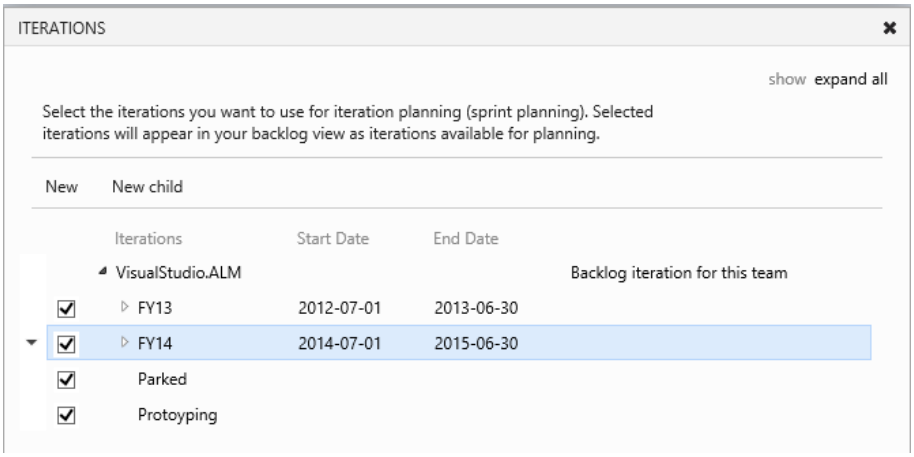
ALM Rangers are ruled by one Backlog.

Enjoying the "visibility" advantages

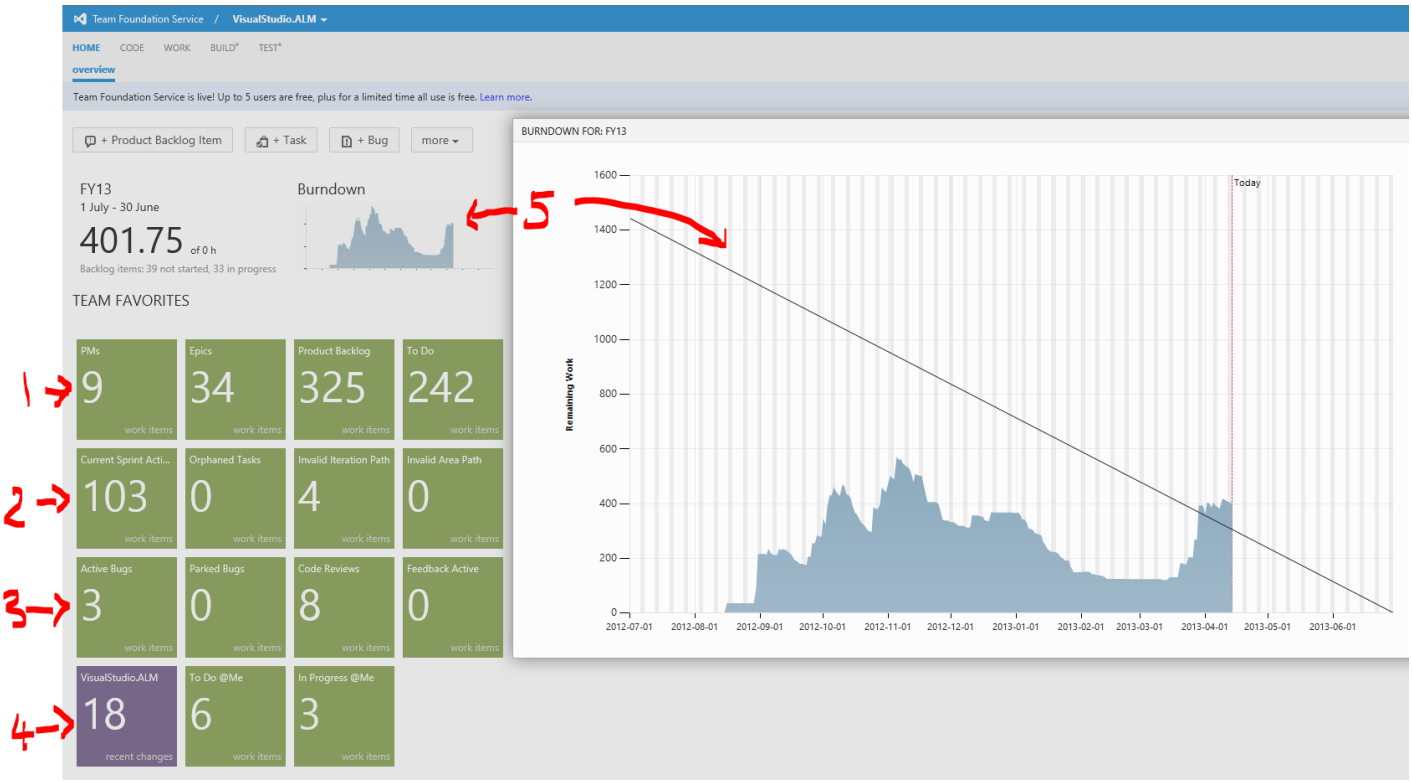
The main advantages are visibility, transparency, and agile planning using the task board, the Kanban features, the default charts, and a number of WIQL queries.

High-Altitude (Program Management) View

Our default Team Project Team is called Core Support Team. Its area is set as the Team project root and iterations to the current financial year (FY13). It is the default page used by the Program Managers and the Ruck Masters coordinating the Ranger teams.



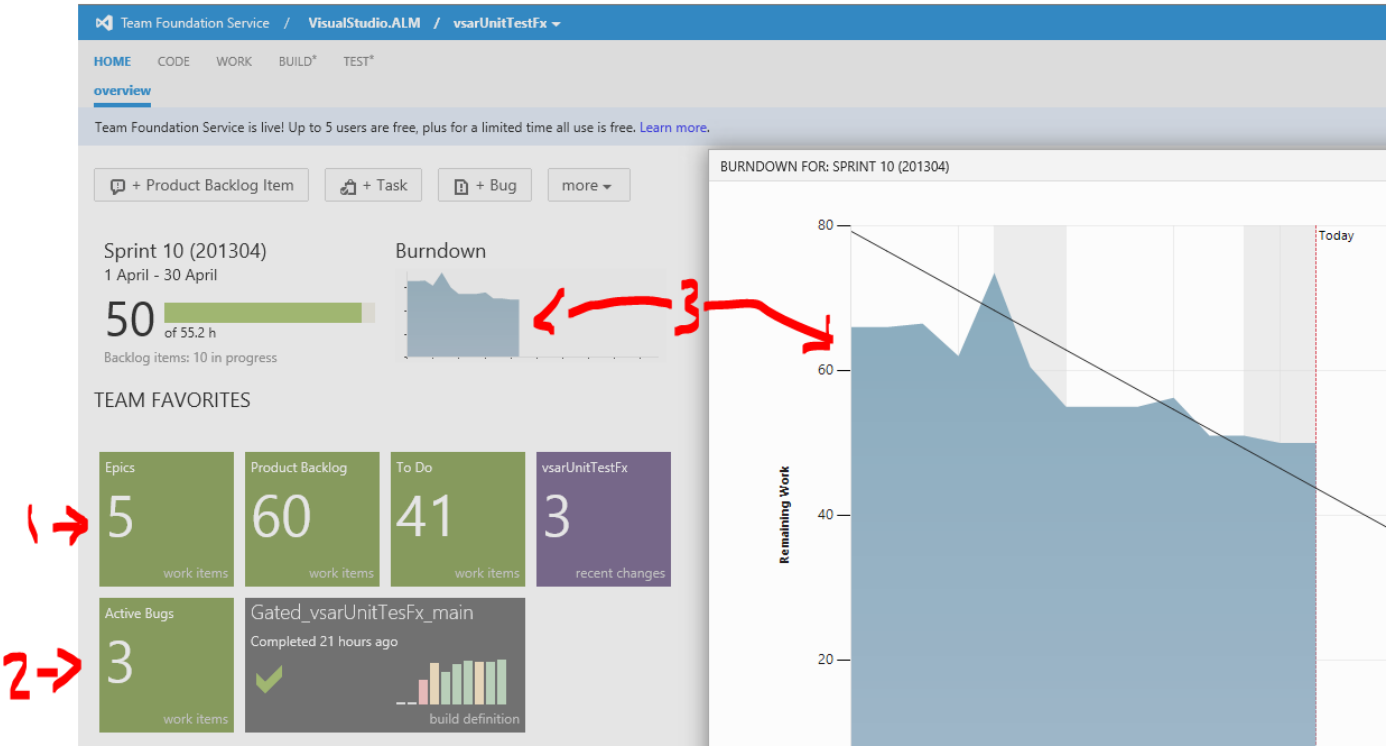
The Core Support Team dashboard gives a "quick glance" overview of the "state of the nation," as shown below:



1. Number of outstanding Program Management, Epics, Product Backlog, and Tasks items that are actively worked on.
2. Anomalies, such as orphaned tasks that have been forgotten in previous iterations, tasks with an invalid iteration or area path. Examples of these WIQL queries are included in the appendix.
3. Active bugs, parked bugs, code reviews, and active feedback requests, which indicate overall quality.
4. Recent version control changes and also tasks that are assigned to ourselves (@Me) and are being worked on. Graphical representation of the aggregated task view of all teams across the backlog, allowing us to identify patterns and, more importantly, "red flag" conditions. The aggregated view will definitely raise a few critical eye brows and the blood pressure of the "burn-down" zealots. It does, however, give us an opportunity to spot patterns and a sense of how we are doing across the financial year. For example …
    - Even though we have four triages and opportunities to start waves of projects, as discussed, we can spot three waves during FY13.
    - The third wave, based on triage 3 and 4, has pushed us above the ideal trend line for the financial year and we are relying on the heroes amongst the Rangers to do the usual sprint towards the finish line.

Low-Altitude (Team) View

When we switch to a specific Team Foundation Team, we drill down to the relevant area path:



1. Epics, Product Backlog and Task items, and version control are Team specific.
2. Active bugs and Build Summaries are Team specific.
3. Burn-down graphs are team and iteration specific.

It is important to highlight that these dashboards, charts, and most of the WIQL queries are standard out-of-the-box.

ALM Rangers have benefited immensely from the Team Foundation Service and its out-of-the-box ALM productivity features.

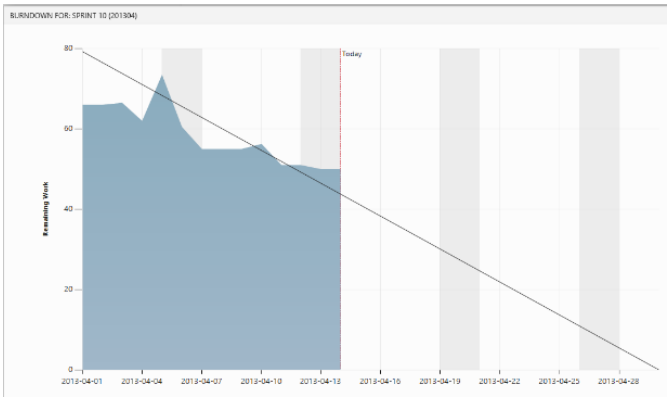
Surely it cannot all be a Rose garden?

GIGO

The GIGO (garbage in garbage out) concept unfortunately applies to this ecosystem. Tasks with incorrect or missing iteration or area paths often vanish in a backlog black hole, creating great confusion and anxiety with ALM Rangers. To identify these anomalies early we are using specialised WIQL queries, such as the invalid area path and orphaned tasks, as mentioned before.

Depending on the size of the GIGO mountain, the core support team and Ruck Masters clean up the backlog when they notice anomalies, or mentor the relevant project leads to clean up their backlog.

Flat, flat, flat … dive, dive, dive!



The part-time nature of the ALM Rangers projects often leads to a burn~~down~~across chart, which suddenly dips and goes into a steep dive. The main reason is that ALM Rangers visit the team task board weekly, rather than daily, and tasks are often committed and closed when done, rather than slowly decremented in terms of remaining time and closed when there is no more work to do. While we typically achieve our sprint objectives and reach the zero (0) remaining work for the iteration, the Rangers burn down pattern can be nerve-racking for all stakeholders.

Do we really need all the history and clutter?

One of the current dogfooding excursions is focused on branching strategies and the need for history. We recently switched from a **Basic** (two branch) to a **Main**-only branching strategy as we found that the need for isolation rarely existed and forward integration (FI) from Main to Dev branches usually came up with no changes.

Other active investigations include:

- When we delete artefacts like Dev branches, do we delete or destroy? Do we really need the history?
- When we move a project from service mode to active, do we move it to our operational Team Project and keep the history in the ServiceMode Team Project, or do we nuke the latter?
- When we perform code reviews, do we do them in parallel with new development or do we block until code reviews are done?

Planned investigations:

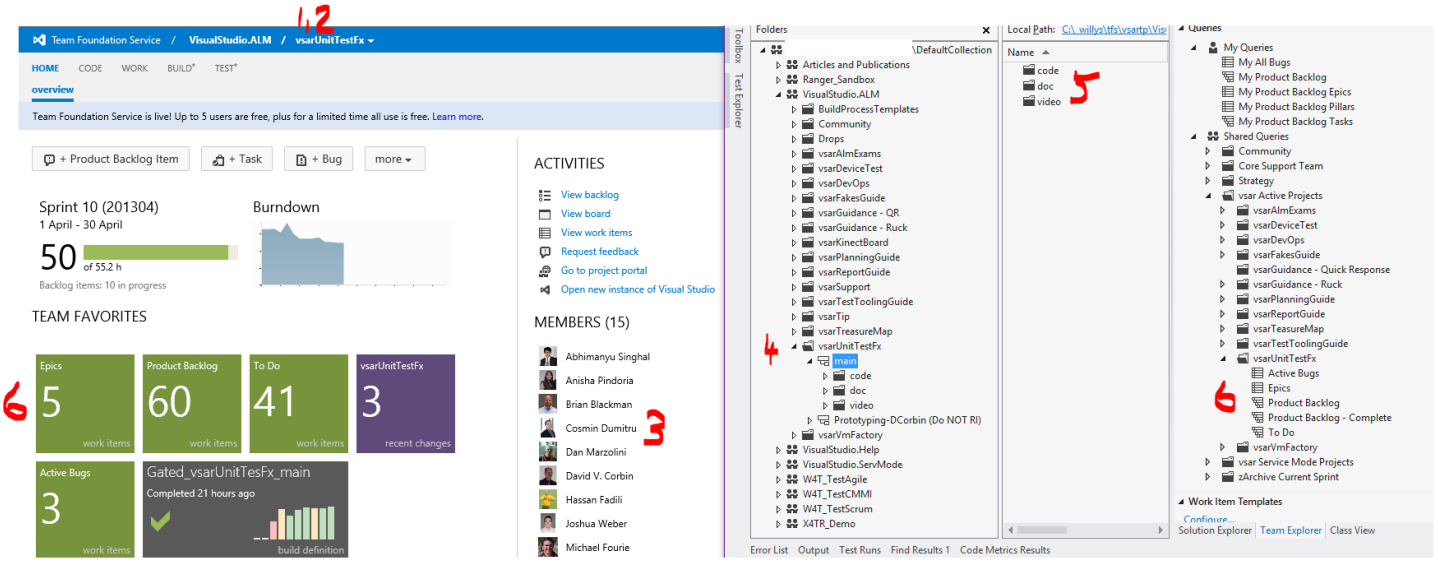
- When we use Test Plans, how do we define them and maintain them across iterations and project versions?
- How do we define and use automated builds for Windows Store Applications?

Ruck Masters mentor and guide the ALM Ranger teams to implement and use Ruck.

*The **continuous learning** and **feedback** ecosystem continuously improves the process and the environment, and most importantly introduces consistency.*

What happens when a project "kick-off meeting" was a success?

To conclude, let's summarise the steps we perform in our environment when a project kick-off meeting was successful and the team gets the "GO GO GO" instruction:



1. **Assign a code** to the project, such as vsarUnitTestFx, which stands for Visual Studio ALM Ranger Unit Test Framework.
2. Create a Team Foundation **Team** and give it the same name as the code.
3. **Assign** Team **Members**, including the product owner, project lead, Ruck master, contributors, and reviewers.
4. Create a **V**ersion **C**ontrol node and give it the same name as the code.
5. Create a **branching strategy** using the **ALM Ranger Quick Response Solutions** TFS Branch Tool.
6. Create **WIQL queries** and select the team favourites, which in turn will light up on the dashboard.
7. Select the **iterations** applicable to the team, ranging from 3-4 month time boxes.
8. Define the **Epics**, then break them down into **Product Backlog** Items and associated **Tasks**.
9. Create a **PM – PBI** that defines the sprint objectives, the definition of done, acceptance criteria, and other reference information needed by the Program Manager. This is the first PBI to be committed and usually the last to be closed. See [Definition of "**DONE**" and knowing when it is safe to sleep](#) [14] for example definitions of DONE.

10. Perform a **sprint planning** session, allocate backlog items to the sprint, agree on a weekly or bi-weekly stand-up meeting and then GO, GO, GO.

Hope this overview has been interesting and that it will enable you to use the Team Foundation Service to create a productive environment for your teams.

Appendix

Example WIQLs

Invalid Iteration Path

```
SELECT  [System.Id], [System.WorkItemType], [System.Title], [System.AssignedTo], [System.State],
         [System.AreaPath], [System.IterationPath] FROM WorkItems
WHERE   [System.TeamProject] = @project and [System.WorkItemType] = 'Task' and
        [System.State] &lt;&gt; 'Done' and [System.State] &lt;&gt; 'Removed' and
        [System.IterationPath] = 'VisualStudio.ALM'
ORDER BY [System.Id]
```

Invalid Area Path

```
SELECT  [System.Id], [System.WorkItemType], [System.Title], [System.AssignedTo], [System.State],
        [System.AreaPath], [System.IterationPath] FROM WorkItems
WHERE   [System.TeamProject] = @project  AND  [System.WorkItemType] = 'Task'  AND
        ( [System.State] = 'In Progress'  OR  [System.State] = 'Done' ) AND
        [System.AreaPath] = 'VisualStudio.ALM'
ORDER BY [System.Id]
```

Orphaned Tasks

```
SELECT  [System.Id], [System.AreaPath], [System.WorkItemType], [System.Title], [System.AssignedTo],
        [System.State], [Microsoft.VSTS.Scheduling.RemainingWork]
FROM    WorkItems
WHERE   [System.TeamProject] = @project and ([System.WorkItemType] = 'Bug' or
        [System.WorkItemType] = 'Task')and [System.State] <> 'Removed' and [System.State] <> 'Done' and
        ([System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 1' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 2' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 3' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 4' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 5' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 6 (Low Bandwidth)' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 7' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 8' or
         [System.IterationPath] under 'VisualStudio.ALM\FY13\Sprint 9 (201303)')
ORDER BY [System.AreaPath]
```

For more information …

See the two blogs blogs.msdn.com/b/visualstudioalm and blogs.msdn.com/b/willy-peter_schaub, as well as the references listed under the footnotes.

Your candid feedback will be most appreciated.

Special thanks to fellow ALM Rangers who helped me compile and improve this adventure

Brian Blackman[15], Michael Fourie[16], Patricia Wagner, William Bartholomew

Footnotes

[1] **Understanding Visual Studio ALM Rangers**
http://aka.ms/vsarunderstand

[2] **ALM Guidance Visual Studio ALM Rangers — Reflections on Virtual Teams**
http://msdn.microsoft.com/magazine/hh394152.aspx

[3] **Team Foundation Server Planning Guide**
http://vsarplanningguide.codeplex.com

[4] **Team Foundation Service**
http://tfs.visualstudio.com

[5] **Migrating from an On-Premises Team Foundation Server to Team Foundation Service Preview Using the TFS Integration Tools**
http://msdn.microsoft.com/en-us/magazine/jj130558.aspx

[6] **ALM Rangers switching to single TPC/TP and multiple Teams model**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/04/28/alm-rangers-switching-to-single-tpc-tp-and-multiple-teams-model.aspx
[7] **ALM Rangers Ruck Process**
http://blogs.msdn.com/b/willy-peter_schaub/archive/tags/ruck+process/
http://vsarguidance.codeplex.com/
[8] **ALM Ranger Solutions**
http://aka.ms/vsarsolutions
[9] **ALM Ranger Quick Response Solutions**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/08/10/toc-quick-response-solutions.aspx
[10] **User Voice**
http://visualstudio.uservoice.com/forums/121579-visual-studio
[11] **How the ALM Rangers are using Visual Studio ALM ... in a practical nutshell**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2013/02/23/how-the-alm-rangers-are-using-visual-studio-alm-in-a-practical-nutshell.aspx
[12] **ALM Rangers ALM Dogfooding and the "Angst" Factor – Part 3**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/08/31/alm-rangers-alm-dogfooding-and-the-angst-factor-part-3.aspx
[13] **ALM Rangers ALM Dogfooding ... dealing with bugs on tfs.visualstudio.com – Part 8**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/12/05/alm-rangers-alm-dogfooding-dealing-with-bugs-on-tfs-visualstudio-com-part-8.aspx
[14] **Definition of "DONE" and knowing when it is safe to sleep**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2013/03/21/definition-of-done-and-knowing-when-it-is-safe-to-sleep-peacefully.aspx
[15] **Brian Blackman**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2010/06/24/introducing-the-visual-studio-alm-rangers-brian-blackman.aspx
[16] **Mike Fourie**
http://blogs.msdn.com/b/willy-peter_schaub/archive/2010/09/28/introducing-the-visual-studio-alm-rangers-michael-fourie-also-known-as-mike.aspx