**ADS Agile Team Management Guide**

**Description:**

This document (doc20470) captures our Agile management approach to decomposing traceable user themes/requirements into executable development team tasks; and our plan for promoting the tasks into the working prototype.

**Agile Management Kick-off in 5 Steps:**

1. Upload this guide and the customer requirements to TeamForge. Decompose the requirements into Themes/Epics & User Stories/Use Cases. Write User Stories in the form: As a <personae> I want <function> so that I can <context>
2. Create a Planning/Backlog folder. The product backlog is the highest-level planning folder, and contains Epics & User Stories that will span several releases, or are not part of a particular release. To enable the burndown for each product, add start and end dates. Prioritize & associate their parent-child relationship. Anyone can add backlog tasks and defects to the Planning folder for Product Owner prioritization and potential inclusion in future releases.
3. Create a Release\_<number> backlog folder within the Planning folder. Each Product can have several release planning folders. Each Release should contain all user stories (possibly epics). Prioritize & associate their parent-child relationship. Each Release planning folder should be mapped to a software release in the GitHub Tag tool. To enable the burndown for each release, add start and end dates. Assign User Stories to the Release folder based on Product Owner priority and desired release theme. Estimate (educated guestimate) effort for User Stories.
4. Create an Iteration/Sprint backlog within the Release folder. Each Release can have several iterations. Each Iteration should contain the User Stories & tasks that will be worked on during the course of the iteration. Use Dev Team inputs (complexity, completion time) to add items to the Iteration backlog and associate them with their parent User Stories. The Dev Team independently estimates, assigns, and updates task status, allowing for parallelism.
5. Work within an iteration: daily stand-up meeting to discuss impediments/work completed/current work & velocity/remaining sprint effort, all team members update their estimated and remaining effort values for their assigned tasks daily and the actual effort upon completion. Add documentation (SOP, Community Emails, Test Scripts, Release Reports, Retrospectives, Tagged Releases, etc.) to the TeamForge Document tab and ‘associate’ (not add as dependency) the documents with their user stories. Demonstrate tested functionality. Determine task promotion worthiness by assessing the Definition of Done and updating the ‘Fixed in Release’ tag. Tag the Release and update its metadata parameter settings to Maturity=Generally Available, for example, and notify users. Reassign incomplete tasks and actionable items for follow-on iterations. Perform Dev Team Iteration Retrospectives. Go to Step 1 or 2.

**Prototype Dependency & Modification Management:**

A 'normal version' number MUST take the form X.Y.Z where X, Y, and Z are non-negative integers, and MUST NOT contain leading zeroes. X is the major version, Y is the minor version, and Z is the patch version. Each element MUST increase numerically. For instance: 1.9.0 -> 1.10.0 -> 1.11.0. Bug fixes (an internal change that fixes incorrect behavior) not affecting the API increment the patch version, backwards compatible API additions/changes (addition to a public API; public API marked as deprecated; new non-breaking functionality) increment the minor version, and backwards incompatible API changes (public API signature/data type modification/removal) increment the major version. Once a versioned package has been released, the contents of that version MUST NOT be modified. Any modifications MUST be released as a new version. A pre-release version (unstable; may not satisfy 'normal version' compatibility requirements) MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version.

**Backlog Flow (top-to-bottom prioritized order; User Stories are captured in the form As a <personae> I want <function> so that I can <context>):**

1. SOW Rqmt [Attachment E] - Theme / Epic (top-level must reference a requirement id / doc)  
   -> a. User Story 1 (est. story points)  
   ->-> i. Backlog Task or Defect 1 (Definition of Done) (est. completion hours) (assignee)  
   ->->-> Sub-tasks for Task/Defect 1 if any (optional Definition of Done) (assignee)  
   ->-> ii. Backlog Task 2 (Definition of Done) (est. completion hours) (assignee)  
   ->-> iii. Backlog Defect 1 (Definition of Done) (est. completion hours) (assignee)  
   >->-> Sub-tasks for Defect 1 if any (optional Definition of Done) (assignee)  
   -> b. User Story 2 (est. story points)

**Use Case Personae – interacts with/triggers our system; our system integrating with/triggering an external system**

* **Administrator – can directly access data within the data stores (DB, flat files, etc.), build and configure the system**
* **Browser User – use the browser to initiate and receive requests to/from the system**
* **Service Provider – our system exposes RESTful web services data access & event triggers for consumers**
* **Service Client – our system consumes RESTful web services from an external source provider**
* **Project Leader – manages the project recourses (people, processes, technologies)**

**Backlog:**

1. Assign One Leader for Prototype Quality: Assign one leader, give that person authority and responsibility, and hold that person accountable for the quality of the prototype submitted. Ref Attachment E Requirement 1 in (doc20470)
   1. Allow or Deny the Promotion of the Prototype: As the main Project Leader, I want to allow or deny the promotion of the prototype, so that I ensure that only prototype components that satisfy requirements get promoted from design through production.
      1. TeamForge contributor permissions: Configure TeamForge contributor permissions for marking iteration and release tasks as complete (DoD: have a non project owner attempt to change the status of an iteration or release task as complete and inspect that it does not allow the change; have a project owner make the change and see it succeed) (.5) (William)
      2. Configure GitHub Contributor Permissions: Configure GitHub project contributor permissions so that only a project owner can write to the master branch (DoD: have a non project owner attempt to push code to the master branch and inspect that it fails; have a non project owner attempt a pull request to the master branch and ensure that it fails; have an owner perform a pull request from integration to the master branch and ensure that it can succeed) (.5) (William)
2. Assemble a Multidisciplinary Team: As a Project Leader, I want to assemble a multidisciplinary and collaborative team that includes at a minimum, two of the labor categories limited to the Development Pool labor categories, so that I can develop the prototype as quoted in Attachment E. Ref Attachment E Requirement 2 in (doc20470)
   1. Identify Resumes for Labor Categories: Identify resumes for the requirements in Attachment C (DoD: at least one resume per labor category is entered into the document management system for this effort) (4) (LaToya)
   2. Allocate Hours for Resources: Allocate prototype development hours for identified labor categories (DoD: Accounting analysis reports and approval documents signature pages for the use of these resources) (2) (LaToya)
   3. Assign Resources for Prototype: Assign in-house resources to satisfy the tasks listed in Attachment C (DoD documentation of the agreement to assist with this effort) (1) (LaToya)
3. Modern & Open Source Technologies: As a Project Leader, I want to use at least five modern and open-source technologies, regardless of architectural layer (frontend, backend, etc.), so that I can build a product that meets requirements at minimal cost. Ref Attachment E Requirement 3 in (doc20470)
   1. Data Call for Technologies: Perform a data call to the prototype personnel to identify technology candidates to implement the prototype requirements (DoD: documentation of the environment technologies, versions, descriptions, prototype use, and identification of technologies to be demonstrated; demonstration of the selected technologies in use by the prototype)
      1. Analyze Requirements for Technologies: Analyze the requirements for technology criteria and employee resumes for possible technology skills in-house. (1) (William)
      2. Execute Technology Data Call: Propose technologies and solicit feedback from company resources for technologies that will satisfy requirements. (1) (William)
      3. Technology Selection for Prototype: Select and create the final draft of the technologies to be used within the prototype. (1) (William)
   2. AngualarJS Front-end Technology: Use AngularJS as the front-end user interface for the web application
      1. Display Items from Application: Implement a display functionality to list the results returned from the application (DoD: inspect the front-end code and associated libraries) (4) (Frank)
      2. Update Items in Application: Implement an edit functionality to manipulate data returned from the application (DoD: inspect the front-end code and associated libraries) (3) (Frank)
      3. Delete Items from Application: Implement a delete functionality to remove data from the database (DoD: inspect the front-end code and associated libraries) (1) (Frank)
      4. Create Item in Application: Implement a create functionality to add data to the MySQL Database instance (DoD: inspect the front-end code and associated libraries) (1) (Frank)
4. PaaS Deployment: As an Administrator, I want to deploy the prototype on a Platform as a Service (PaaS) provider and indicated which provider used, so that I can utilize the industry proven resource scalability features of the platform. Ref Attachment E Requirement 4 in (doc20470)
   1. Create AWS EC2 Template: Spin-up an Amazon Web Services (AWS) Elastic Computing (EC2) prototype environment template to install and Configure the technologies identified in the Technologies Identification List for this prototype (DoD: The template is stored in Graham Technology’s content management system)
      1. Install and Configure Ubuntu OS: Install and Configure Ubuntu OS as the prototype host operating system (1) (Brian)
      2. Install and Configure Eclipse: Install and Configure Eclipse as the developer IDE (1) (Brian)
      3. Install and Configure Tomcat: Install and Configure Tomcat to host the prototype (DoD: able to view the Tomcat home screen within a browser) (1) (Brian)
      4. Install and Configure Apache Web Server: Install and Configure Apache Web Server to route requests to Tomcat (1) (Brian)
      5. Install and Configure Java JDK: Install and Configure Java JDK for coding the prototype (1) (Brian)
      6. Install and Configure MySQL: Install and Configure MySQL to store prototype data (1) (Brian)
      7. Install and Configure Jenkins CI: Install and Configure Jenkins to monitor and move the prototype to hosting environments (1) (Brian)
      8. Install and Configure Git: Install and Configure Git to perform CM on the source code and associated artifacts (1) (Brian)
      9. Install and Configure Maven: Install and Configure Maven to build, package, and deploy the prototype (DoD: builds and deploys the application to the application hosting container) (1) (Brian)
      10. Install and Configure Monit: Install and Configure Monit to monitor the prototype environment (DoD: monitors Continuous Integration and Application Server tools) (1) (Brian)
   2. AWS Integration Environment Provision: Spin-up an AWS integration environment using the prototype template (DoD: The AWS instance is accessible via a browser and the technologies are verified by inspection) (3) (Brian)
   3. AWS Production Environment Provision: Spin-up an AWS production environment using the prototype template (DoD: The AWS instance is accessible via a browser and each technology can be viewed within Monit or determined to be active via a demonstration) (3) (Brian)
5. Unit Test Code: As an Administrator, I want to write unit tests for the prototype code, so that I may test new and legacy functionality in an automated fashion prior to promoting the code through the development lifecycle. Ref Attachment E Requirement 5 in (doc20470)
   1. Test Read functionality of the REST Service for the external data consumption: Write unit tests to test the functionality of the internally exposed REST services that reads from the external REST service and returns items (DoD: inspect unit test reports indicate success/failure of test runs for returned results of the exposed REST service and is returned an HTTP Status of 200 for successful request/responses only) (2) (Rodney)
   2. Test Read functionality of the REST Service for the Local DB: Write unit tests to test the functionality of the exposed REST services that reads from the database and returns DB items (DoD: inspect unit test reports indicate success/failure of test runs for received requests to an exposed REST service and is returned an HTTP Status of 200 for successful request/responses only) (2) (Rodney)
   3. Test Create functionality of the REST Service for the Local DB: Write unit tests to test the functionality of the exposed REST services that inserts data into the database (DoD: inspect unit test reports indicate success/failure of test runs for inserting and receiving a an HTTP Status of 200 for successful request/responses only) (1) (Rodney)
   4. Test Update functionality of the REST Service for the Local DB: Write unit tests to test the functionality of the exposed REST services that updates data in the local database (DoD: inspect unit test reports indicate success/failure of test runs for updating and receiving a an HTTP Status of 200 for successful request/responses only) (1) (Rodney)
   5. Test Delete functionality of the REST Service for the Local DB: Write unit tests to test the functionality of the exposed REST services that delete data in the local database (DoD: inspect unit test reports indicate success/failure of test runs for updating and receiving a an HTTP Status of 200 for successful request/responses only) (2) (Rodney)
6. Continuous Integration: As an Administrator, I want to set-up or use a Continuous Integration (CI) system to automate the running of unit tests and continuously deploy code to the PaaS provider, so that I can reduce deployment time from development to integration for successfully tested code. Ref Attachment E Requirement 6 in (doc20470)
   1. Configure the CI Tool for Automation: Configure the Jenkins CI tool to monitor code commits to GitHub, and promote the build if it passes predefined criteria (DoD: demonstrate that Jenkins monitors the Configuration Management (CM) tool to build and deploy the application from the CM tool to an application hosting environments) (2) (Brian)
7. Configuration Management: As an Administrator, I want to use Configuration Management (CM) on my prototype, so that I can recover my prototype to a working state in case the current prototype is unusable or undesirable. Ref Attachment E Requirement 7 in (doc20470)
   1. Create GitHub Project: Create a GitHub project for the CM of prototype code (DoD: screenshot of the project home page)
      1. Invite GitHub Contributors: Invite project contributors to join the GitHub project (DoD: screenshot of the project contributors) (.5) (William)
      2. Assign GitHub Contributor Permissions: Assign GitHub roles and responsibilities for the prototype contributors (DoD: screenshot of the project contributor roles) (.5) (William)
      3. Create GitHub Branches: Create a master and integration branch for promoted development code (DoD: screenshot of the project structure) (.5) (William)
   2. Create a TeamForge Project: Create a TeamForge Project for the CM of Agile process products (requirements, user stories, etc.)
      1. Invite TeamForge Contributors: Invite project contributors to join the TeamForge project (DoD: screenshot of the project contributors) (.5) (William)
      2. Assign TeamForge Contributor Permissions: Assign TeamForge roles and responsibilities for the prototype contributors (DoD: screenshot of the project contributor roles) (.5) (William)
      3. Create Team Forge Agile Folder Structure: Create the planning, release, and iteration folder structure (DoD: screenshot of the project structure) (.5) (William)
      4. Post TeamForge Documentation: Post associated documents (requirements, project plan, etc.) to the documents library (.5) (William)
   3. Configure IDEs with GitHub: Configure developer Integrated Development Environments (IDE) to push/pull to/from the prototype GitHub project (DoD: Screenshot of Git Repository view within the IDE) (.5) (Rodney, Frank)
8. Continuous Monitoring: As an Administrator, I want to use Continuous Monitoring of the prototype environment, so that I can receive alerts when a component of the prototype is not online or is non-responsive. Ref Attachment E Requirement 8 in (doc20470)
   1. Configure Continuous Monitoring UI: Configure Monit to monitor services and display the status of the services in a user interface (DoD: view via demonstration of the Monit user interface the services status; receive an alert/log) (2) (Brian)
   2. Configure Continuous Monitoring Alerts: Configure Monit to send alerts or log when a monitored service is not online or responsive (DoD: stop Tomcat and verify that Monit notifies via email or logs) (2) (Brian)
9. Software Container Deployment: As an Administrator, I want to deploy the software in a container (i.e., utilized operating-system-level virtualization), so that I can access the prototype outside of the hosting container via the World Wide Web. Ref Attachment E Requirement 9 in (doc20470)
   1. Configure Maven Deployment Automation: Configure Maven to deploy a WAR package of the code to the Tomcat server (DoD: upon failure of a build, verify via demonstration that a new WAR file is not created within Tomcat; upon a successful build verify that a WAR file is deployed to the Tomcat server and unpacked) (1) (Rodney)
10. REST Consumer and Provider: As an Administrator, I want to make use of an Application Programming Interface (API), by either consuming or providing one RESTfully. so that I can provide or consume data in a RESTful manner with Browser User, Service Provider, and Service Consumer personas. Ref Attachment E Requirement 10 in (doc20470)
    1. Consume FedData REST Service: Consume the external FedData Web Service interface and display the results to the browser (DoD: within a browser, execute the application functionality to consume the external RESTful service and ensure that FedData is returned to the browser and/or stored in the internal database via inspection) (2) (Rodney)
    2. Provide Create Locally REST Service: Expose a RESTful Web Service to create data within the internal database (DoD: demonstrate that a RESTful consumer can send a request to the application via a browser and/or a machine to add data, and be returned an indication of a successful or unsuccessful database manipulation attempt; and inspect the database for the changes) (2) (Rodney)
    3. Provide Read Locally REST Service: expose a RESTful Web Service to read data from the internal database (DoD: demonstrate that a RESTful consumer can send a request to the application via a browser and/or a machine, and be returned an indication of a successful or unsuccessful database manipulation attempt; and inspect the database to verify the values) (2) (Rodney)
    4. Provide Update Locally REST Service: Update expose a RESTful Web Service to update data within the internal database (DoD: demonstrate that a RESTful consumer can send a request to the application via a browser and/or a machine, manipulate data, and be returned an indication of a successful or unsuccessful database manipulation attempt; and inspect the database for the changes) (2) (Rodney)
    5. Provide Delete Locally REST Service: expose a RESTful Web Service to delete data within the internal database (DoD: demonstrate that a RESTful consumer can send a request to the application via a browser and/or a machine to delete data, and be returned an indication of a successful or unsuccessful database manipulation attempt; and inspect the database for the changes) (2) (Rodney)
11. Detail Agile Scrum Approach with Feedback Loops: As a Leader, I want to use an iterative approach, where feedback informs subsequent work or versions of the prototype, so that I may improve quality using the lessons learned. Ref Attachment E Requirement 11 in (doc20470)
    1. Configure TeamForge Agile Management System: Configure TeamForge Agile management system to document and guide the prototype development iterations (release and sprint backlogs), and iteration demonstration and retrospective documentation (DoD: inspect TeamForge backlogs, burn-down charts, demo notes/action items, retrospective documentation) (3) (William)
12. Create Prototype Documentation: As an Administrator, I want to be provided sufficient documentation to install and run the prototype on another machine, so that I can ensure a working product with repeatable process on the same or similar technologies. Ref Attachment E Requirement 12 in (doc20470)
    1. Create Prototype ReadMe: Create a readme file that will live alongside the prototype within GitHub with instructions to run the application on another machine (DoD: demonstrate the deployment of the system using the steps in the readme file) (2) (William, Rodney, Brian, Frank)
13. Open License & Free of Charge Technologies: As a Leader, I want to ensure that the prototype and underlying platforms used to create and run the prototype are openly licensed and free of charge, so that I may implement a prototype that meets requirements at minimal cost and avoid copyright/legal technicalities. Ref Attachment E Requirement 13 in (doc20470)
    1. Document All Technologies and their License Status: Document all technologies used, their purpose, and their licensing status (DoD: inspection of the technologies document) (2) (Brian, Rodney, Frank)