Paper SD04

# Re-Inventing the Deliverable Tracker - from Excel to Dashboard

Megan Harries, Veramed, London, UK

## ABSTRACT

Deliverable Trackers are a critical tool in biometrics workflows and are used for everything from project planning, resource management, through to QC and validation tracking, and as a quality audit record. The standard approach is to use Microsoft Excel– this offers a well understood user interface, however it relies on manual entry that is often out of date, and data is duplicated in multiple places with limited ability to link and synchronise data from other systems.

This presentation demonstrates a new approach to implementing a deliverable tracker that uses an interactive user-interface backed with a database populated by harvesting metrics directly from the SCE and from other systems via API calls. This approach provides delivery teams with a recognisable deliverable tracker user experience however significantly reduces the need for manual entry, provides live automated results and also allows project metrics to be pooled centrally to enable data-driven process improvement insights.

## INTRODUCTION

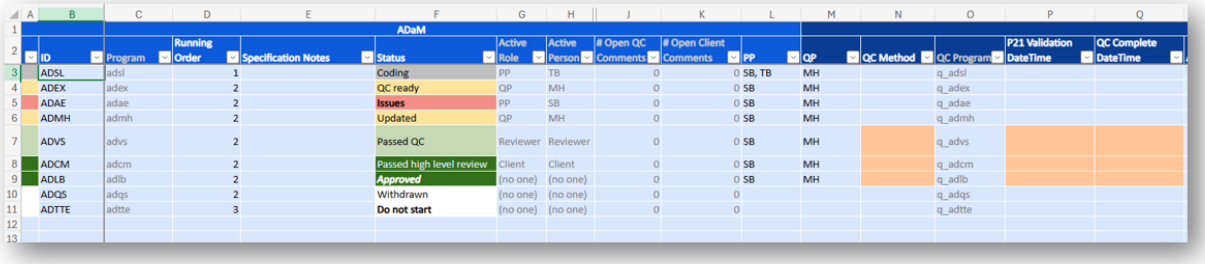
Deliverable trackers are foundational tools in statistical programming workflows, supporting everything from project planning and resource allocation to quality control and audit readiness. Traditionally, the industry has relied on Excel trackers to monitor the lifecycle of deliverables due to their accessibility, often incorporating bespoke functionality tailored to individual sponsors and Contract Research Organisations (CROs). While this approach has previously been considered a standard practice, it presents several limitations and opportunities for improvement. As the use of modern Statistical Computing Environments (SCEs) and the demand for real-time insights increase, the industry is shifting towards more dynamic, integrated solutions.

This paper discusses the benefits and drawbacks of several types of deliverable trackers and introduces a modern approach to tracking through a flexible, custom-built dashboard backed by a database and API integrations. The dashboard replicates the familiar user experience of traditional trackers while automating data collection from SCEs and version control platforms such as GitHub. It supports custom workflows, user assignments and status transitions, whilst also abstracting Git operations to reduce the learning curve and repetitive burden on study teams. With built-in metrics, audit trails and system validations this solution enhances transparency, collaboration and operation efficiency across statistical programming reams.

## AVAILABLE TRACKERS

### EXCEL DELIVERABLE TRACKER

Although Excel based trackers have long been the industry standard, there is a growing shift of moving away from these types of trackers towards more modern solutions.



| | ADaM | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ID | Program | Running Order | Specification Notes | Status | Active Role | Active Person | # Open QC Comments | # Open Client Comments | PP | QP | QC Method | QC Program | P21 Validation DateTime | QC Complete DateTime |
| 3 | ADSL | adsl | 1 | | Coding | PP | TB | 0 | 0 | SB, TB | MH | | q_adsl | | |
| 4 | ADEX | adex | 2 | | QC ready | QP | MH | 0 | 0 | SB | MH | | q_adex | | |
| 5 | ADAE | adae | 2 | | Issues | PP | SB | 0 | 0 | SB | MH | | q_adae | | |
| 6 | ADMH | admh | 2 | | Updated | QP | MH | 0 | 0 | SB | MH | | q_admh | | |
| 7 | ADVS | advs | 2 | | Passed QC | Reviewer | Reviewer | 0 | 0 | SB | MH | | q_advs | | |
| 8 | ADCM | adcm | 2 | | Passed high level review | Client | Client | 0 | 0 | SB | MH | | q_adcm | | |
| 9 | ADLB | adlb | 2 | | Approved | (no one) | (no one) | 0 | 0 | SB | MH | | q_adlb | | |
| 10 | ADQS | adqs | 2 | | Withdrawn | (no one) | (no one) | 0 | 0 | | | | q_adqs | | |
| 11 | ADTTE | adtte | 3 | | Do not start | (no one) | (no one) | 0 | 0 | | | | q_adtte | | |

*Figure 1 Example of a Deliverable Tracker in Excel*

Excel deliverable trackers do offer many advantages:

- **Familiarity:** Widely used across the industry, requiring minimal training
- **Visual status tracking:** Colour coding allows quick identification of deliverable status across a study

- **Offline accessibility:** Trackers can be used offline if required
- **Customisation:** Columns can be easily added or adapted to meet sponsor specific needs
- **TMF Compatibility:** Data can be exported quickly for inclusion in the Trial Master File (TMF)
- **Quick setup:** Once a template is established, new trackers can be created rapidly.

However, there are notable limitations to this approach:

- **Manual updates:** Frequent manual input is required, increasing the risk of errors
- **Limited integration:** Excel does not easily connect with external systems and platforms
- **Version control challenges:** Offline use can lead to un-synced changes and confusion with a team
- **Scalability:** Complex formulas in a file-based tracker can make trackers difficult to manage and upgrade
- **Filtering:** Users may not to be able to view all assignments if these are across multiple development streams
- **Tabular format:** Multiple rounds of QC can result in cluttered and confusing comments columns
- **Assignment management:** Handling multiple contributors within a single deliverable can be challenging.

#### PROJECT MANAGEMENT TOOLS FOR DELIVERABLE TRACKING

Utilising project management tools, such as Jira, for tracking the lifecycle of a deliverable is becoming increasingly popular within the industry. These tools offer a range of benefits tailored to task and workflow management, however, also bring some downfalls. Many of these drawbacks occur as tools such as Jira were primarily developed for use in software development, often for tasks with a less rigid structure than statistical programming.

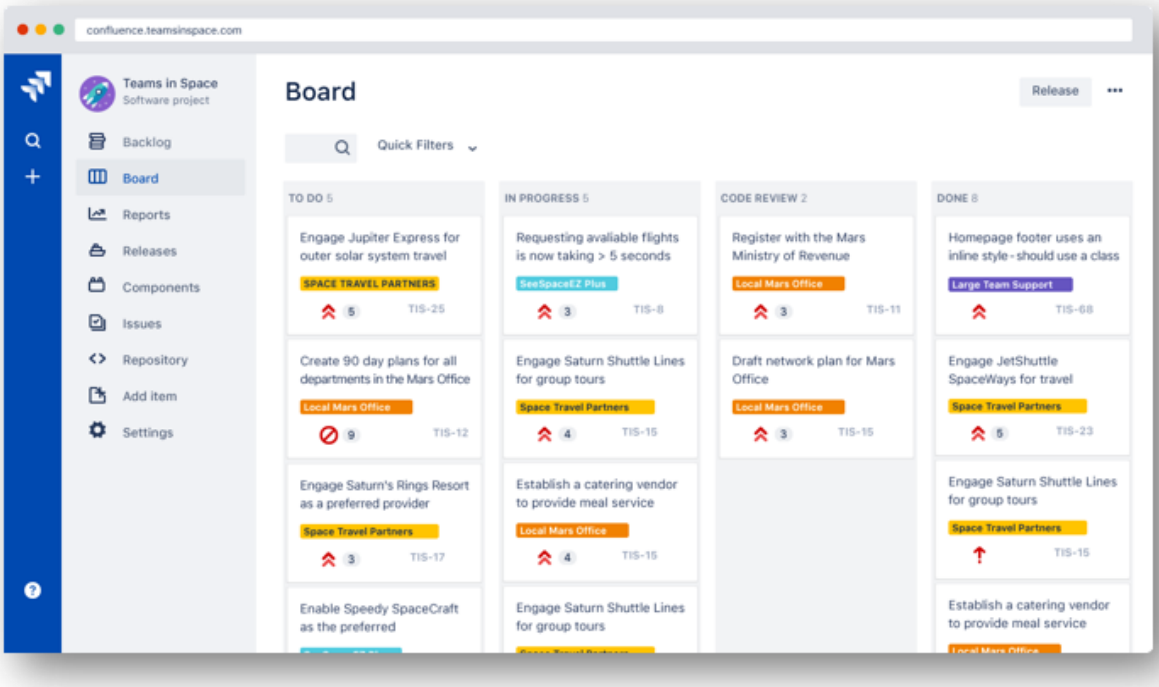



*Figure 2 Example of a project management tool*

Some of the benefits of project management tools include:

- **Purpose built for task management:** Clear ownership of tasks with assignees, due dates and priorities
- **Visual workflow:** Drag and drop interfaces enable intuitive progress tracking and support customised workflows for different deliverable types
- **Enhanced collaboration:** Users can add comments, tag team members and link related deliverables for better communication
- **Dependency tracking:** Default features allow easy identification and management of task dependencies
- **Audit trails:** All changes are logged, providing full traceability
- **Customisation:** Custom fields and automation rules can be configured to suit specific project needs
- **Tool integration:** Compatible with Git platforms and some SCEs
- **Filtering:** Filters can be set to view work across multiple projects or programming streams

- **Time-tracking:** Start and end dates can be populated automatically to monitor task duration and identify overdue items
- **Performance metrics:** Insights into programming timelines and workload assignment

Some of the drawbacks include:

- **Initial setup effort:** Defining workflows, fields and permissions can be time consuming
- **Task creation overhead:** More detailed input is required beyond just naming the deliverable
- **Learning curve:** Teams unfamiliar with the tool may underutilise useful features such as comments for asynchronous and visible communication
- **Complexity for small teams:** May introduce unnecessary overhead for small teams, bringing in the risk of spending more time managing the tool than the work
- **Licensing costs:** These tools can be expensive, especially for large teams and enterprise features
- **Partial adoption risks:** Incomplete usage could lead to parallel maintenance of trackers or QC comments logs elsewhere
- **TMF integration:** Additional steps may be needed to transfer information into the Trial Master File.

#### DELIVERABLE TRACKER DASHBOARD

Utilising dashboards for deliverable tracking offers a modern, improved alternative to traditional methods. How a deliverable tracking dashboard can be implemented will be discussed further and the main topic of this paper.

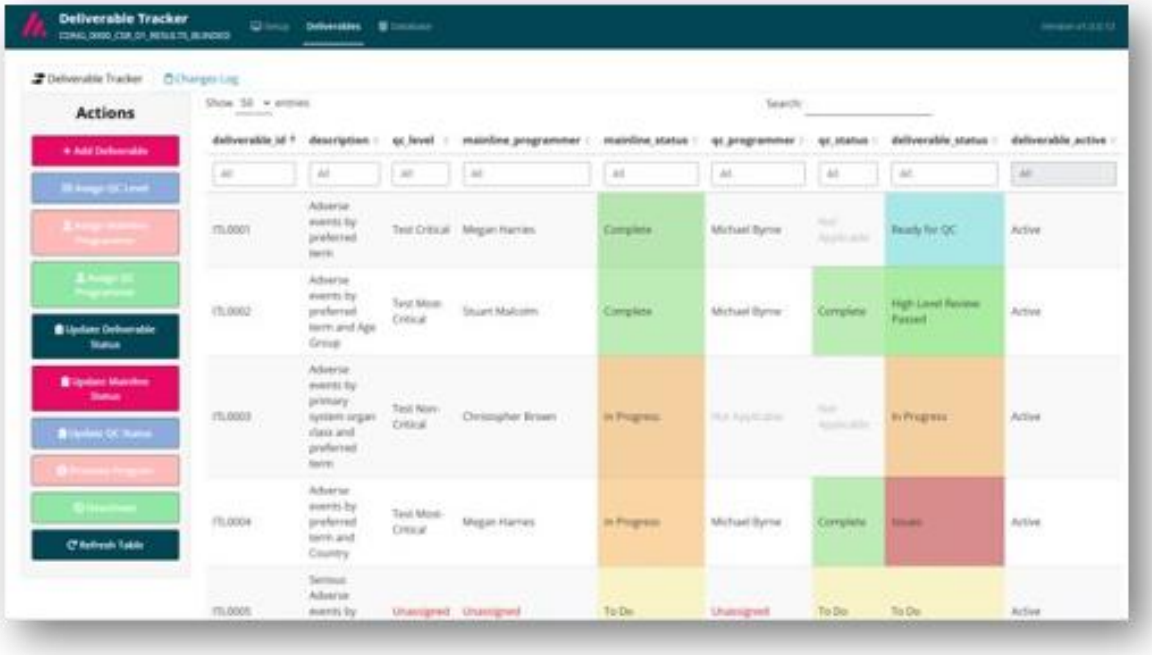



*Figure 3 Example of a Deliverable Tracker Dashboard*

The key benefits to utilising a dashboard for deliverable tracking include:

- **Real-time updates:** Automatically reflects progress and QC status directly from the SCE
- **Enhanced metrics:** Supports additional insights such as programming time, QC coverage and timeline risk indicators
- **Automation:** Minimises manual effort and reduces the risk of human error
- **Audit trail:** Provides clear traceability of actions, showing who did what and when
- **Custom visualisation:** Interactive charts and filters to support informed decision making
- **Centralised view:** Ensures all team members are working from the same data, reducing miscommunication

There are drawbacks to all solutions, some of the cases where dashboards do not perform as well as other options include:

- **Initial Setup:** Requires development time and may involve infrastructure changes

- **Training needs:** Team members may need additional time onboarding to use the dashboard effectively
- **System dependency:** If the SCE or hosting platform experiences downtime, dashboard access will be limited
- **Maintenance overhead:** Dashboards require ongoing updates and support to remain effective.

## DASHBOARD FEATURES

#### IMPORT OF DELIVERABLE LIST

To begin tracking deliverables within the dashboard they must first be imported in, typically from an externally created list. The dashboard example we developed to show this use case can read in any .csv or .xlsx file. Upon upload, the file is previewed for the user, who can then select specific columns to map to required fields in the tracker such as deliverable ID, deliverable description etc. Once this file is imported, the tracker is populated with the relevant data, marking remaining fields as a default option such as 'Unassigned' or 'To Do' ready for the user to begin tracking the deliverable.

In addition to the initial bulk import, this dashboard also allows manual entry, as new deliverables might be required after the initial import. Deliverables can also be 'deactivated', preserving their metadata at the time of deactivation while removing them from active tracking, edits and downstream system usage. If needed, deactivated deliverables can be reactivated at any time.

Upon import of deliverables, whether manually or through a file upload, checks can be introduced to ensure that standard naming conventions for deliverables are used if needed. If the same set of deliverables is required for a different tracker, then functionality can be included to ensure these trackers mirror the deliverable set used and update accordingly.

#### CUSTOM ACTIONS

One of the significant benefits of using a dashboard approach is the ability to define and execute a wide range of custom actions. These actions can be presented as interactive buttons through a sidebar panel, as shown in Figure 3. Once a deliverable is selected these interactive elements allow users to perform specific tasks, the actions can be restricted based on the user's role permissions if the user's details are passed to the dashboard.

For example, users can:

- Assign workflows tailored to the deliverable type or risk
- Allocate programmers to specific tasks
- Update statuses based on progress or QC outcomes
- Promote programs through different stages of the workflow.

#### GIT ABSTRACTION

One of the key benefits of assigning a workflow to a deliverable in the dashboard is the ability to abstract Git operations. As many statistical programmers and study teams may be unfamiliar with Git, this abstraction helps reduce the learning curve when implementing Git-based version control within the SCE as the dashboard can handle some Git interactions for the user.

In our implementation of the dashboard tracker, users can select from the available workflows defined in the system. Upon selection, the necessary branches are created automatically to facilitate the chosen workflow. Standard naming conventions can be applied and adhered to through the automation of this process, reducing human error and ensuring that downstream processes can rely on the naming conventions being followed. This type of abstraction is helpful for the user both for reducing the learning curve and saving time in completing repetitive tasks.


*Figure 4 Git abstraction through the promotion of code to produce relevant Pull Requests in GitHub*

Our dashboard also automatically created and abstracted out Pull Requests (PRs) upon promotion of a program through the workflow. The user simply indicates the level the code is currently on in the workflow and which level the code needs to be promoted to. The system generates the PR and assigns the relevant user as an approver within GitHub.

This integration is powered by the GitHub API, which facilitates communication between GitHub and the dashboard for branch creation, PR creation and user account linkage. One of the major advantages of dashboards is the ability to have this integration with external systems, supporting the industry's growing shift towards a broader adoption of Git for version control.

**ASSIGNMENT OF USERS**

Assigning programmers to deliverables and enabling them to view their assignments is a core function of any deliverable tracker. In the dashboard we created, user assignment is streamlined through an interactive drop-down list of names or user IDs, populated using metadata from the SCE to only display users with programming access to the selected project. This approach helps prevent common issues like assigning programmers who aren't part of the project, misspelled names or confusion caused by duplicate initials within the project team.

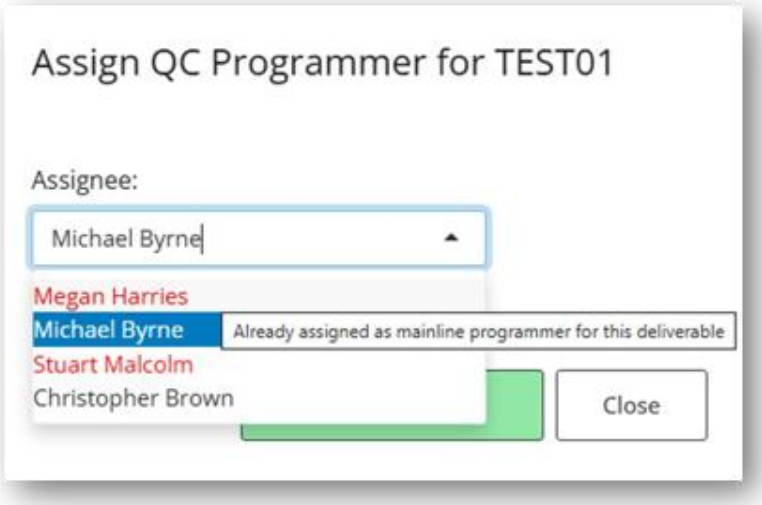



*Figure 5 Assignment of users flagging if they have already been assigned to a previous programming stream*

One of the benefits of this method is the ability to flag potential conflicts, such as assigning a programmer to a deliverable they previously QC'ed (or vice versa). This level of validation is difficult to achieve in traditional Excel trackers, which typically allow only one assignee and lack automation to prevent such conflicts.

**ASSIGNMENT OF STATUSES**

One of the next key uses of a deliverable tracker is the ability to view and update the status of each deliverable, or the individual tasks required for the deliverable to progress through its lifecycle. In traditional Excel trackers, status assignment is often visualised using colour coding for quick reference. While dashboards can replicate this visual clarity, they also offer enhanced functionality to facilitate process workflows.

Process workflows can be incorporated so that status transitions follow a logical sequence. For example, a deliverable cannot jump directly from 'To Do' to 'High Level Review Passed', instead the system enforces a progression through intermediate states that are mandated by the workflow. This reduces errors and ensures compliance to existing workflows across teams.

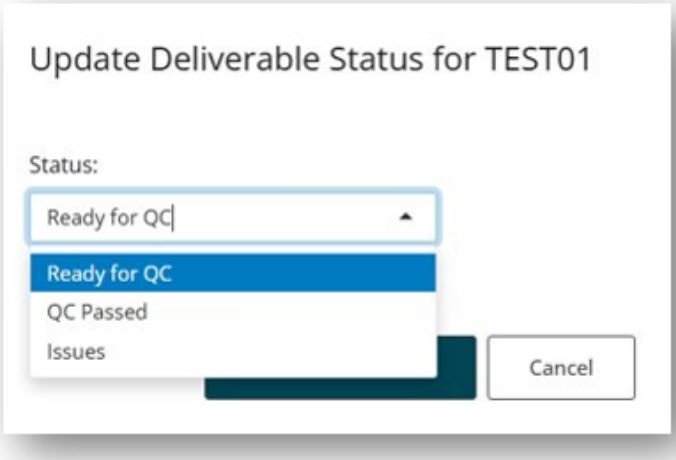



*Figure 6 Restriction of possible deliverable statuses based on current information*

These statuses can also be used as an input for other functions, such as promotion may only be allowed to occur if the deliverable is in the required status. It also is possible for the statuses to be restricted based on the information available in the SCE, for example if there are outstanding QC comments then the deliverable cannot progress to 'QC Passed' or if a program has been found with the same deliverable ID then the status can be updated automatically from 'To Do' to 'In Progress'.

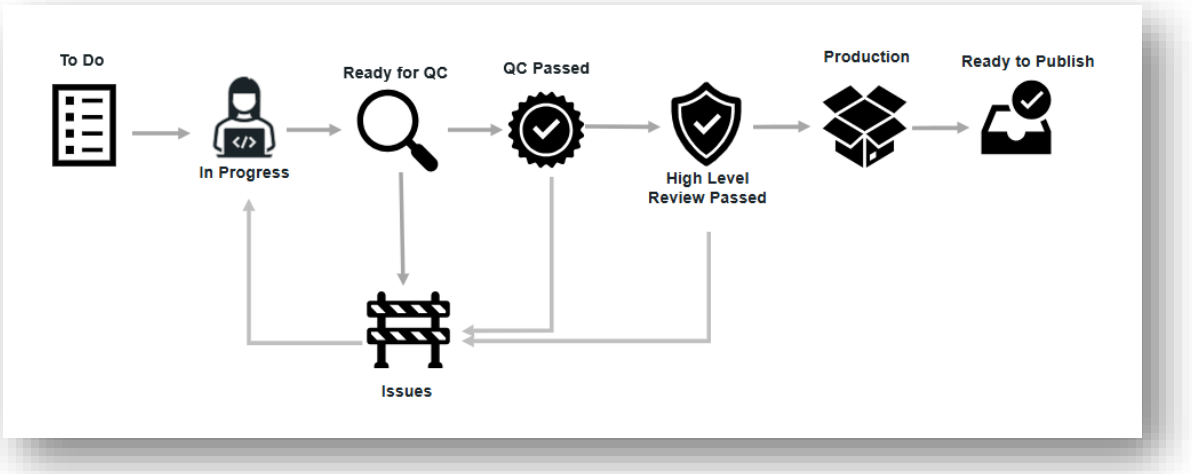



*Figure 7 Progression workflow of deliverables through the available statuses*

**CUSTOM USER MESSAGES**

Custom user messages play an important role in enhancing the usability and transparency of dashboard interactions. Whenever a user performs an action, whether it's updating data or interacting with external systems, the dashboard can display a message detailing the outcome. These messages may include confirmation of updates, system error messages received, links to external systems for easy navigation, or contextual guidance.

These messages are particularly valuable in scenarios where an action is restricted or fails. For example, if a user attempts an invalid operation, such as trying to assign a status to the development of the QC program prior to the QC programmer being assigned, the dashboard can provide a clear explanation and instructions on how to proceed. Similarly, if a connection to an external system (such as GitHub or the SCE) fails, an appropriate error message is displayed to help the user understand and troubleshoot the issue.

The messages displayed can incorporate different visualisations to differentiate between information, warnings, errors and success messages to guide the user attention appropriately. Messages can also be adjusted to either be temporary or persistent depending on the severity and if time needs to be allowed for users to navigate to external links or for error messages to be investigated.

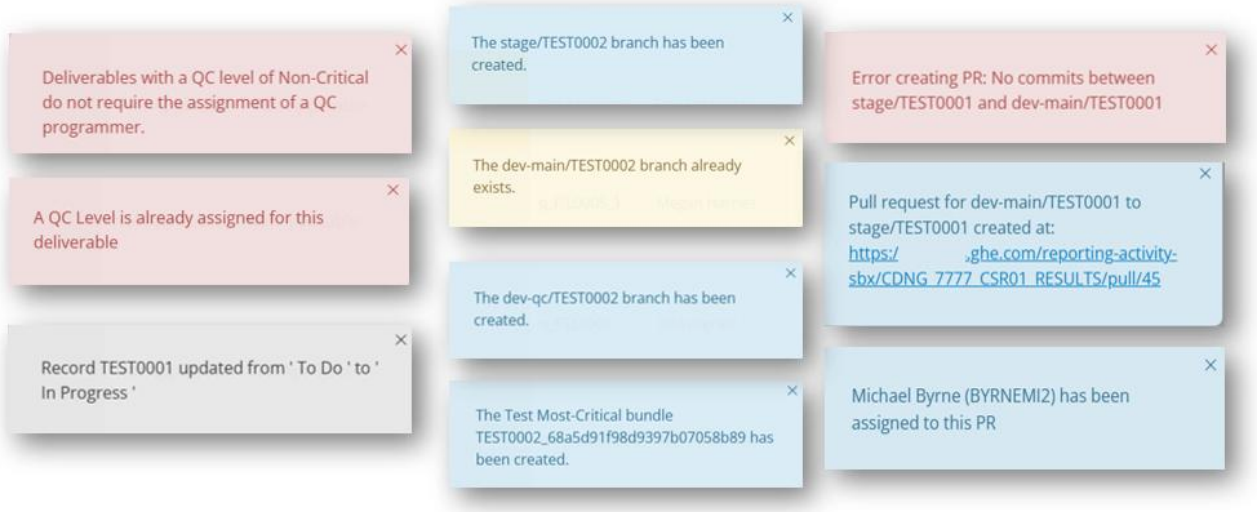



*Figure 8 Examples of custom user messages to provide both successful and unsuccessful notifications*

**METRICS**

One of the notable advantages of using a dashboard for deliverable tracking is the ability to leverage recorded data to provide useful and bespoke metrics tailored for study leads or resourcing teams. These metrics provide valuable insights that support decision making and improve operational efficiency. Dashboard metrics can be customised to tailor metrics views based on the user's role, e.g resource manager, study lead or programmer.

Some examples of additional metrics that can be displayed within the dashboard are:

- Visualising the status of deliverables across the study to monitor overall development progress
- Resource management to understand team member assignments both within and across studies to better manage workloads and identify capacity gaps
- Forecasting and planning to use historical data to predict future workloads and resource needs for similar projects
- QC coverage metrics to track how many deliverables have undergone QC and the outcome of those reviews
- Reporting activity metrics, to analyse the progress of different types of deliverables and identify any bottlenecks that may impact downstream tasks.

**LOGGING ACTIONS**

A key advantage for using a dashboard for deliverable tracking is the ability to log all changes made within the system. This provides a transparent audit trail, allowing team members to review the full history of a deliverable's lifecycle. Each logged action can be associated with a specific user, if the dashboard facilitates this, and timestamped to indicate exactly when the change occurred.

The metadata management for the dashboard we utilised a PostgreSQL database to store the deliverable tracking data and the logging data. Access to this database is managed via a service account, ensuring that individual user credentials are not required. This setup enhances security and ensures that data can only be accessed or modified through the dashboard interface, maintaining integrity and control.

## CONCLUSION

The dashboard-based deliverable tracker represents a significant advancement from the current way of working through using Excel for deliverable tracking. Through integrating directly with external platforms such as GitHub or the SCE, it enables automation of repetitive tasks, enforces workflow compliance and ensures consistent, real-time visibility across teams.

Excel often has version control issues with multiple copies of trackers being available at the same time, a dashboard ensures that the same view of the deliverable metadata is seen by everyone, removing the risk of viewing and making decisions from outdated information. Excel trackers can also become difficult to maintain if used across multiple projects, and if additional functionality is required to be included then it can become a mammoth task to update all versions of the trackers. These drawbacks are countered with the dashboard approach, through the ease of managing deployments across all teams using the dashboard with testing incorporated to reduce the risk of incompatibility.

There is no doubt a steeper learning curve for users to migrate from current ways of working to a dashboard approach, alongside the development time required for initial implementation and maintenance. However, the long-term benefits are substantial. Teams gain access to a tool which is designed for its purpose and makes daily tasks more streamlined, while study leads benefit from centralised, accurate metadata which drives better decision making. This approach not only brings efficiency gains in deliverable tracking, but also lays the foundation for scalable, data-driven process improvement bringing an enhanced way of working.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged.  Contact the author at:

**Megan Harries**
Veramed Limited
Email: megan.harries@veramed.co.uk