Metrics Dashboards for Agile Development of Safety-Critical Software

9/27/2023 |

Kelly Gasperski, P.Eng.
Software Product Assurance, Canadarm3
KELLY.GASPERSKI@MDA.SPACE

Software Metrics

- What are software metrics?
  - Data about the software development process or software products being produced.
  - **Progress** (% work completed), **Predictive** (estimates vs actuals), **Quality** (# defects, % tests passing), and **Development Experience** (# of broken builds, time to run tests)

- Why do we collect software metrics?
  - To comply with safety-critical program requirements and standards, such as as NPR 7150.2D and ECSS-Q-ST-80C Rev 1.
  - To assess adherence to the Software Quality Model defined for the program.
  - To capture and identify trends that may affect the overall success of the program.

Software Development

- Canadarm3 is the first major program to pave the way for Agile development and CI pipelines at MDA.
- Using Table 5-3 in ECSS-Q-HB-80-04A as a guideline, there are a number of metrics we are required to collect and trend for our safety-critical software applications, such as:
  - Adherence to Coding Standards
  - Code Coverage, Nesting Levels, and Cyclomatic Complexity Limits
  - Detection of outdated software libraries and cybersecurity vulnerabilities

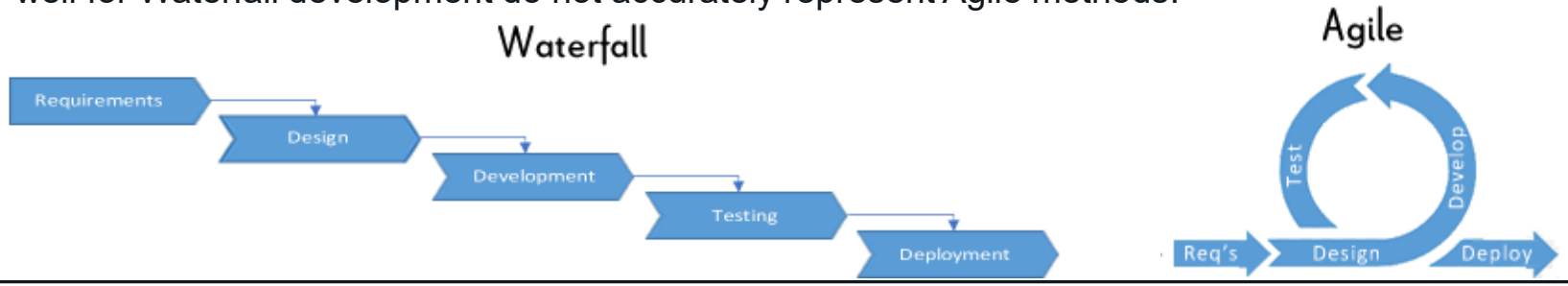
- In addition, metrics relating to the software development process and environment are also collected.

Scaled Agile

- Canadarm3 software development follows a Scaled Agile approach, with multiple software development teams working in parallel on a common code base.

- Metrics relating to the quality of the code, technical debt accrued, and the average velocity of the development teams allows us to identify issues early and plan future work accordingly.

- This required tailoring our set of software metrics for Agile development, as certain metrics that work well for Waterfall development do not accurately represent Agile methods.



Continuous Integration (CI) Pipelines

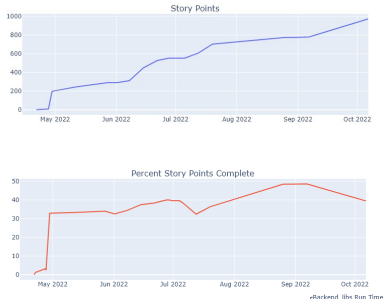
- CI Pipelines help enforce our safety-critical program requirements by ensuring all code meets the defined quality checks and thresholds.

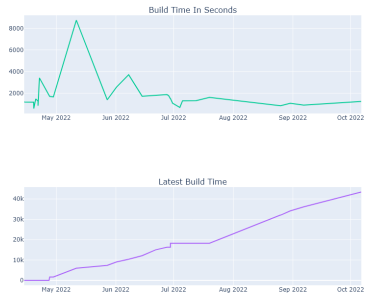
- This creates a valuable set of data we can collect, to identify issues in both the software being produced as well as the software development environment itself.

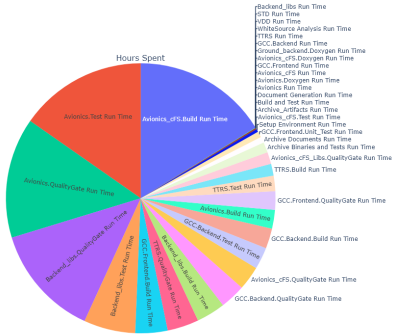
- Pipeline health monitoring allows us to understand where bottlenecks in the development process may occur, leading to improvements and optimizations in our code and infrastructure.

Making Sense of the Data Collected

- Developers, DevOps, Managers, Customers, and Product Assurance teams could all benefit from the trends and insights that software metrics provide.
- The collection and processing of data can be automated using modern tools, then turned into graphical representations that are easy to interpret with human eyes.
- Next…how do we display these visualizations to maximize their value?







Metrics Dashboards

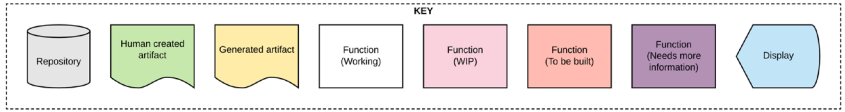
- Data visualizations displayed on monitors throughout the work area.
- All stakeholders can access data quickly and easily.
- Developers and DevOps teams can see real-time feedback.
- Managers and SPA can quickly and easily identify issues to address.
- Acts like an “office campfire” drawing people together, leading to new insights, discussions, and ideas.

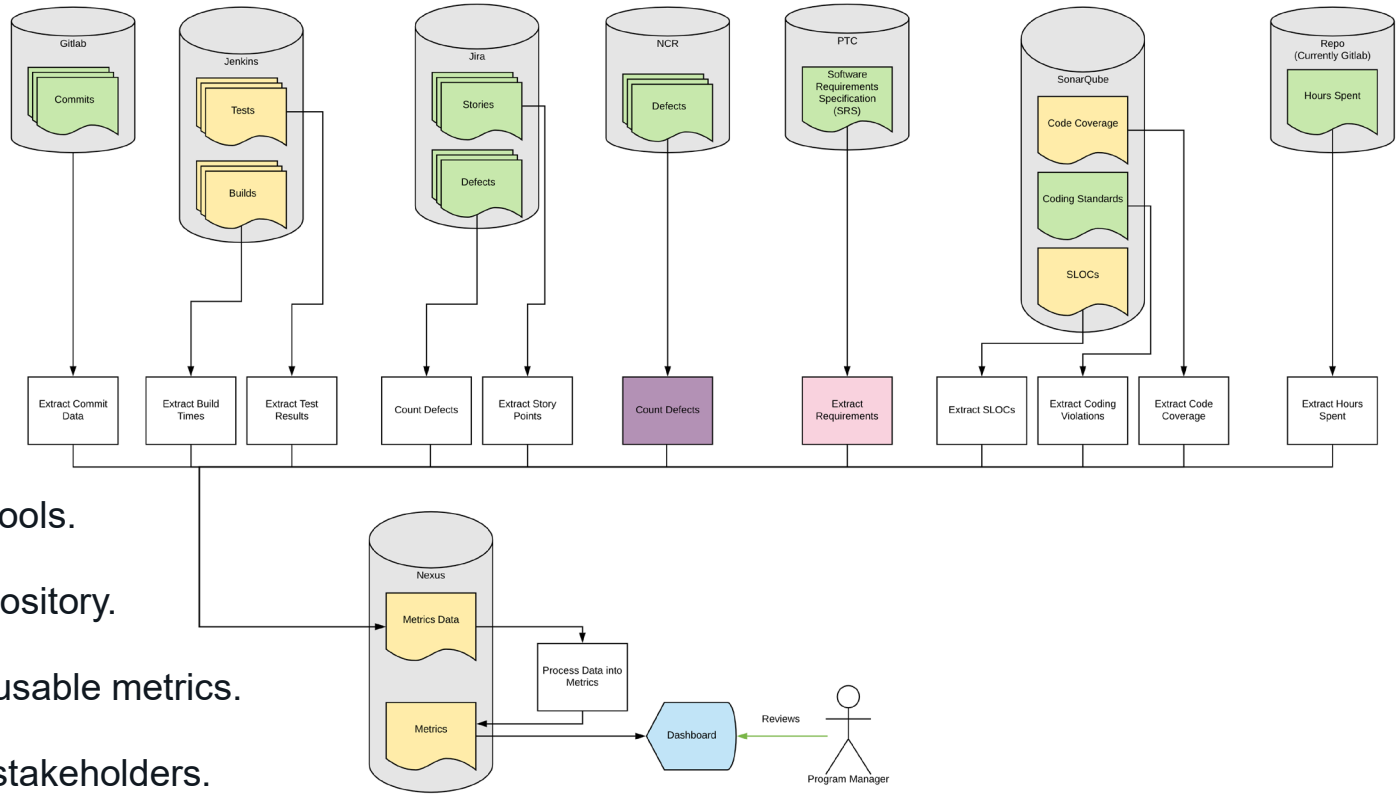
Vision

- Establish a fully automated, customizable, and scalable metrics process that can be reused across multiple projects.
  - Data can be pulled from any source necessary at regular intervals.
  - Data can be presented dynamically at any time.

- New projects simply set their metrics configurations (data to collect, tools to connect to); no additional effort is required to meet their program's metrics requirements.
- Metrics Dashboards displayed throughout the work area provide immediate, real-time feedback on the software development process, environment, and products being produced.

Architecture





1. Extract data from tools.
2. Store data in a repository.
3. Process data into usable metrics.
4. Display to project stakeholders.

Step 1: Extract data from tools

- Two main approaches:
  - Pull the data from the tools by having scripts use their REST APIs.
  - Push the data from the tools with routine exports or direct database connections.

- Examples of data to extract from common tools:

Git

- # of Commits to Main and their pipeline results (Pass/Fail)

Jenkins

- Time to Build, Time to Test, # of Tests Passing, # of Tests Failing/Skipped

Jira

- Completed Story Points, Remaining Story Points, # of Informal Defects

SonarQube

- SLOC, # of Code Smells, Code Coverage, Cyclomatic Complexity

Windchill

- # of Software Requirements, Verification Methods, Safety Criticality Tags

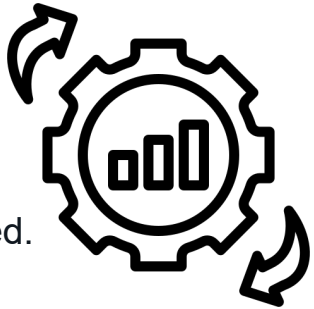
Mend

- # of Outdated Libraries, Licensing Violations, Cybersecurity Vulnerabilities

Step 2: Store data in a repository

- Repository can be as simple as a collection of CSV files, or it can be a shared database.
- Collection mechanisms should continually append data to the repository with corresponding timestamps.
- Repository backups and verification checks should be performed routinely to ensure no gaps in the data occur.

Step 3: Process data into usable metrics



- Scripts or database functions can be used to perform calculations on the collected data to achieve the desired software metrics.
- This step may also be done as part of the visualization layer, depending on the tools selected.
- Examples of post-processed metrics:

Git

- # of Commits to Main and their pipeline results (Pass/Fail)
  - % Commits to Main with failed pipeline runs

Jenkins

- # of Tests Passing, # of Tests Failing/Skipped
  - % Tests Passing

Jira

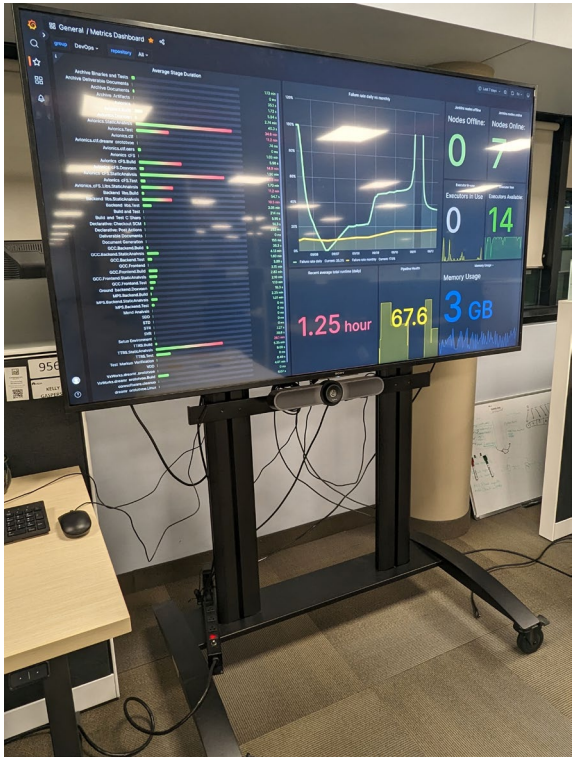
- Completed Story Points, Remaining Story Points
  - % Work Completed

Windchill

- # of Software Requirements, Verification Methods, Safety Criticality Tags
  - % Safety-Critical Software Requirements Verified by Test

Step 4: Display to project stakeholders

- There are many tools available for generating visualizations for the post-processed or raw data, which in turn can be hosted on internal servers and accessed via web browser.
- Tools like Plotly, Grafana, or Power BI can be used to create interactive graphs and customized dashboards tailored to different stakeholders.
- Providing links to these dashboards or displaying them on TVs or monitors around the office allow easy access to valuable information, eliminating the need to request or generate static reports.



- Additional enhancements supported by Metrics Dashboards:
  - Changing colours based on defined thresholds
  - Sending alerts when thresholds are reached
  - Providing interactive graphs, and adjustable time ranges
  - Ability to transition between different dashboards, on-demand or as a slideshow

How to establish your own Metrics Dashboards

1. Define the set of software metrics required.
   - Elicit feedback from stakeholders (Devs, SPA etc.)
   - Tailor to Agile processes, where applicable
2. Set up a data collection and storage mechanism.
   - Scripts using REST APIs, or direct database connections
   - Allow for configuration so that it remains flexible for other projects or changing tools
3. Set up post-processing calculations and visualizations for the metrics.
   - Turn the raw data into usable metrics, then display those in easy-to-interpret graphs and charts
4. Display the visualizations in prominent locations around the office.
   - Watch as stakeholders gather and discuss, leading to new insights, ideas, and innovations!

Questions?

THANK YOU

CONTACT INFO:

Kelly Gasperski, P.Eng.
Software Product Assurance, Canadarm3

KELLY.GASPERSKI@MDA.SPACE

MDA