Clinical Decision Support (CDS) Testing Process

Elisa Dell'Oglio, MSBE; Ronelle Stevens, PharmD; Charles Lagor MD, PhD; Susan Smith; Karen Bavuso RN, MSN

Clinical Informatics, Partners eCare, Partners HealthCare System, Boston, MA

Keywords: Knowledge Management, KM, Clinical Decision Support, CDS, Content Testing, Requirements Testing

Introduction/Background

Clinical decision support (CDS) has become the hallmark of an electronic health record (EHR). There is an increasing priority on rule and alert implementation in the setting of Meaningful Use, clinical safety, mitigation of complex practice decisions, and other organizational initiatives. At Partners Healthcare, the Knowledge Management (KM) team, a subgroup of Clinical informatics, was tasked with authoring test procedures to ensure that the design intentions of customized CDS interventions to be used in a vendor-based enterprise EHR were preserved. The aim of this project was to create a process to facilitate and standardize authoring of test procedures based on design/build specifications and track pass/fail progress of CDS requirements.

Methods

We developed a process to author, test, and track the status of CDS requirements by leveraging Microsoft Visual Studio[1] testing tools. CDS interventions design and build specifications were tracked in Jira[2] and programmatically uploaded as requirements within Team Foundation Server (TFS)[3]. Workflow demonstrations were provided by Application Teams. A library of reusable shared testing steps was created within the Microsoft Test Manager (MTM)[4] testing tool based on validated provider workflows established within the vendor based enterprise EHR. Test case parameters including test patient's instances were defined within each test case's iteration. Test script authoring and execution was carried out by separate Knowledge Engineers (KEs) in MTM. A QA process was established in MTM for independent review of test scripts prior to testing execution to decrease script bias.

Results

The goal was to perform testing of approximately 500 CDS interventions. Each of the CDS interventions contained approximately 3 test scripts and 1-6 iterations per script. Shared test steps helped streamline authorship and decreased fragmented testing language. Testing efficiency was optimized by performing in-depth configuration testing first, followed by shallow testing and iterations through different parameters values of the build. We assigned a one-time-use reserved test patient to each test case/iteration to avoid testing bias. Testing progress was captured through the standardized use of testing statuses at the individual requirements level and related bugs level when failure occurred. Additionally, the process was designed to accommodate searchable handoff between test planners, test executors and the application team(s) involved in troubleshooting build when testing failure occurred.

Discussion/Conclusion

Designing a testing process for CDS was challenging. In an effort to ensure thorough testing and validation of CDS interventions, it is key to establish a structured process to plan, test, and track the progress of CDS requirements, associated test cases, and progress on bug remediation in the instances where failures occurred. We are continuing to work towards further developing testing processes and testing best practices for CDS.

References

[1] Microsoft Visual Studio 2012 http://www.visualstudio.com/
[2] Atlassian Jira (v5.2.7) https://www.atlassian.com/software/jira
[3] Team Foundation Server 2012 https://msdn.microsoft.com/en-us/library/ms181238(v=vs.90).aspx
[4] Microsoft Test Manager 2012 https://msdn.microsoft.com/en-us/library/jj635157.aspx