Development & Validation Process

Analyse-it for Microsoft Excel

This document describes the development and testing process used by Analyse-it Software, Ltd. in creating Analyse-it for Microsoft Excel version 3.00 and later.

1st March 2020

About Analyse-it

Analyse-it for Microsoft Excel is developed by Analyse-it Software Ltd. It is an add-in for Microsoft Excel that implements statistical procedures and is designed to run on Microsoft Excel 2007, 2010, 2013 and 2016 on Microsoft Windows operating systems Windows Vista, Windows 7, Windows 8, Windows 10, and Windows Server 2003, 2008, 2012 and 2016.

For the last 20 years Analyse-it has been the leader in statistical analysis software for Microsoft Excel. It is used by over 30,000 users, is highly respected in the scientific and research community, and is cited in thousands of peer reviewed publications.

http://analyse-it.com/

Using Analyse-it for Microsoft Excel in FDA regulated environments

For purposes of U.S. Food and Drug Administration (FDA) validation, Analyse-it for Microsoft Excel should be considered *off-the-shelf* software.

For that reason Analyse-it is not directly responsible for compliance with FDA regulations. Responsibility for compliance lies with the user. To that end, you should read this document to assess the adequacy of our software development process and determine what, if any, additional efforts you need to take to establish that the software is validated for the intended use.

For FDA advice on software validation, see:

http://www.fda.gov/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm085281.htm

The Development Process

The development process involves Analyse-it and end-users.

Software Development Tools

Languages	C#, with some numerical libraries in C++ and FORTRAN
Development environment	Microsoft Visual Studio 2017 / 2019
Configuration & source code management	GIT
Requirements, issue, & defect management	Microsoft DevOps
Test case management & test runs	Internal
User Interface prototyping	Balsamiq Mockups
Unit testing	JetBrains ReSharper / NUnit
Code coverage	JetBrains dotCover
Performance & memory-use	JetBrains dotTrace
Help documentation	DITA OT
Software installers	MSI Factory

Software Development Lifecycle

Revisions to the software are scheduled for release on a 9-12-month cycle, or less depending on circumstances and market demands, with minor maintenance releases as necessary:

- Major releases incorporate new statistical procedures and/or features.
- Maintenance releases contain fixes to issues reported by end-users and minor feature enhancements.

Requirements and Design

Requirements are gathered through several channels:

- Customer feedback
- Technical support enquires
- Development discussion and recommendations
- Partner requests

Requirements are tracked using the issue management system in the form of user-stories, each documenting what the user wants to achieve. User interface mockups are developed using the user interface prototyping tool to explore possible implementation.

All requirements are prioritized, among other development tasks, to produce a backlog from which future development can proceed.

Release planning

When preparing for a new release, the most desirable requirements from the backlog are chosen and added into a release backlog. The release backlog is prioritized to ensure the most important requirements, or those with high-risk in terms of development time or complexity, are addressed first.

A specification document is then produced outlining the features for implementation. Implementation is broken into a series of short sprints (stages) with target dates and milestones. Milestones and target dates are not fixed and are regularly reviewed and adjusted during development to ensure code stability, code quality, & testing coverage, remain high and to adjust for delays such as requirements needing clarification.

Coding

Each coding sprint starts by choosing requirements from the release backlog to be coded in the sprint. Coding then proceeds to implement the planned requirements and changes.

During coding:

- All code changes are recorded in the source code management system, with notes where necessary, to provide an audit trail of all changes.
- Automated and manual ad-hoc testing begins, with test cases created and recorded in the test case management system.
- Bugs are recorded in the defect tracking system and fixed immediately when possible to ensure code quality and stability remains high.
- Developers discuss and explain their design and coding choices, and solicit feedback from other developers in the team through code reviews.
- Performance & memory-use analysis tools are used to ensure the implementation is efficient.

When implementation of new features and significant changes are complete, though not fully tested, the requirements for the current release are frozen. Coding sprints now focus on rigorous testing and fixing bugs until the software is stable. At this time approval is needed for further changes, with all code changes reviewed to ensure they address the intended issue only and do not introduce additional errors.

Testing

Throughout development, unit, integration, and system tests are developed to ensure the software is of the highest quality, stability, and is numerically correct. Testing is manual and ad hoc in early development. Automatic tests are developed as the code becomes more stable, and are used whenever possible so they can be automatically re-run to regression test changes and releases.

During testing:

- All test cases, automatic and manual, are documented.
- All code changes are recorded in the source code management system, with notes where necessary, to provide an audit trail of all changes.
- Bugs and their fixes are recorded in the defect tracking system.

In total we maintain a collection of more than 1,000 proprietary test cases for Analyse-it. Almost all tests are multi-faceted, exercising many inputs and testing many outputs.

Exhaustive testing is not possible for anything but the most trivial of software, so testing resources must be concentrated on the most high-risk areas. All statistical functions are tested against other software including *JMP*, *Mathematica*, against in-house developed Excel spreadsheets, and against public validation suites including the NIST StRD (http://www.itl.nist.gov/div898/strd/). Testing ensures the numerical accuracy and correctness of the algorithm implementation. Code reviews are used to identify edge cases and special conditions that need to be tested, and code coverage is measured at the branch/decision level to ensure all-paths through the algorithms are tested.

Despite our extensive testing efforts, it should be noted that testing can never guarantee that software contains no bugs. To quote the FDA:

"Software testing has limitations that must be recognized and considered when planning the testing of a particular software product. Except for the simplest of programs, software cannot be exhaustively tested. Generally it is not feasible to test a software product with all possible inputs, nor is it possible to test all possible data processing paths that can occur during program execution. There is no one type of testing or testing methodology that can ensure a particular software product has been thoroughly tested. Testing of all program functionality does not mean all of the program has been tested. Testing of all of a program's code does not mean all necessary functionality is present in the program. Testing of all program functionality and all program code does not mean the program is 100% correct! Software testing that finds no errors should not be interpreted to mean that errors do not exist in the software product."

http://www.fda.gov/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm085281.htm#_Toc517237962

User Site Testing

Development relies on end-user data and feedback to perform system testing in specific and unique customer environments.

At the end of coding sprints that include significant new features or changes, interim releases are sent to domain experts. Feedback is solicited to identify problems in installation, usability, feature implementation, and to determine whether requirements have been met. Selected end-users are invited into this process as the software nears completion.

During these phases, all feedback and bug reports are recorded in the issue and defect tracking system where they can be evaluated, prioritized, and either scheduled to be addressed in the remaining sprints of the release or deferred to a future release. Automated bug-reporting ensures technical information on bugs/crashes at user sites is collected and relayed directly to the development team to assist diagnosing and fixing any issues that arise.

Release

When development and testing are complete, a production release candidate is provided to select end users. A test-run covering the highest-risk tests is run on the release candidate across different variations of Excel and operating systems. The software is then ready for public release.

Maintenance Releases

After a major release, bugs and issues that arise are collected through customer support and a proprietary bug reporting and feedback feature of the software. All feedback and bug reports are recorded in the issue and defect tracking system where they can be evaluated and prioritized.

When high priority issues arise, that need attention before the next major release, a coding sprint to fix the issue is scheduled. During this sprint all changes to the code are recorded and documented in the source configuration management, and fixes are documented in the defect tracking systems. The fixes are then scheduled for inclusion in the next maintenance release.

Maintenance releases occur as needed. Before a maintenance release, code changes in the maintenance release are reviewed, fixes are tested before inclusion in the maintenance branch, automated tests are updated appropriately, and then tests are re-run to regression test the maintenance release. The documentation team is notified of any changes affecting the documentation.