A Data Science tool for Data Reviewing  
Giuseppe Di Monaco, UCB BioSciences GmbH, Monheim, Germany

Abstract

Data Science arose from the blending of scientific and engineering methods. This new field requires expertise from different areas such as data cleaning, statistics, and software engineering (or IT). Naturally, a significant challenge is associated with obtaining in-depth knowledge in all of these areas. In order to overcome this, each member of our team is assigned to the tasks that best fit their strengths. We developed tools that cover different parts of the areas mentioned above. The architecture of these tools was designed using rigorous techniques for development, testing and validation with a view to enabling fully scalable and automated maintenance. This paper will focus on a tool called "Data Review" that ensures the data is clean and consistent as possible. This tool can either work as an independent application or can be part of a greater project covering more aspects of data science.

Introduction (Problem)

Often reviewers are asked to review data within a short timeframe. Our solution to this challenge is a software application called "Data Reviewer". Our goal is to improve the quality, efficiency and cost-effectiveness of the data reviewing process. This project was implemented using software engineering methodology. This method allows the application to include over more components needed during the review process (for example TPL creation or validation.) The application could also become part of a much greater Data Science project.

Formal method for Software Engineering

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Design</th>
<th>Development</th>
<th>Testing</th>
</tr>
</thead>
</table>
| 1 Define type of user  
2 Define user interface & Reporting  
3 Define system requirements (User requirements expected to be heavy weight process) | 1 Choose the modularity technique.  
2 Identify Patterns and abstraction  
3 Use quality measures | 1 Choose the model based on business needs  
2 Evaluation of changes  
3 Incremental Tests | 1 Unit & Integration tests  
2 Black box and white box testing  
3 Regression testing |

<table>
<thead>
<tr>
<th>(1) Expert Users &amp; Non-Expert users</th>
<th>(2) User Interface &amp; Output Reporting</th>
<th>(3) Modules Components &amp; Application</th>
</tr>
</thead>
</table>
| • Expert Users Have full right to make decision where parameters for the modules.  
• Non-Expert users are provided with a set of guidelines that allow the ability to perform the tools settings. | • The Interface must be clean, concise, consistent, efficient and intuitive.  
• Output Reporting must be effective and well-structured. It should follow the 4 C’s. Clear, correct, complete and correct. | • Modules are portions of a program that can be specified, that can be tested, and that can be changed independently.  
• Components in the Data Reviewer tool are a set of modules.  
• Application on the Data (Metadata) is a list of all the data that needs to be added. The application must be scalable and accept any kind of input data (structured or not). |

<table>
<thead>
<tr>
<th>(1) Modularity</th>
<th>(2) Design Patterns &amp; Abstraction</th>
<th>(3) Cohesion &amp; Coupling</th>
</tr>
</thead>
</table>
| • Modularity is a software design technique which specifies the functionality of a program by modularizing the component of the modularization of modules. Modularity allows easy maintenance while facilitating the level of abstraction. | • Design Patterns are general, reusable solutions to commonly occurring problems in software development.  
• Abstraction also used to identify what function a module should do. For example, a module should not do anything that the user cannot describe.  
• Coupling refers to what degree modules are dependent on each other. Low coupling means that changing something in one module should not affect the others. | • Cohesion is a quality measure of the software modularity. High cohesion means the module is focused on what should be done.  
• Coupling refers to what degree modules are dependent on each other. Low coupling means that changing something in one module should not affect the others. |

<table>
<thead>
<tr>
<th>(1) Incremental Model</th>
<th>(2) Evaluation of changes</th>
<th>(3) Incremental Tests</th>
</tr>
</thead>
</table>
| • Incremental Model is a development method of software engineering where the product is designed and implemented incrementally.  
• Incremental testing is a process of testing changes in the system in small steps.  
• Incremental Tests can be easily performed because few changes are made within any single iteration. This allows for more targeted and rigorous testing of each element within the overall product. | • Evaluation of changes is based on previous feedback. A plan is developed for the next increment and modifications are made accordingly.  
• Incremental testing is the process of testing changes in the system in small steps.  
• Incremental Tests can be easily performed because few changes are made within any single iteration. This allows for more targeted and rigorous testing of each element within the overall product. |

Result (Data Review Application)

Data Review application is made up of five primary components. The sixth component will perform data analysis to assess the applicability of checks for a given indication or study. Other components can be included. Good candidates would be components to perform TPLs creation and/or validation. The entire application with all its components sits on top of a validation platform EMM (Environment Macro Management) which ensures the 4 phases of the Tool’s Lifecycle (Development, Testing, Release, Support) are fully controlled and automated.

<table>
<thead>
<tr>
<th>COMPONENT (I)</th>
<th>COMPONENT (II)</th>
<th>COMPONENT (III)</th>
<th>COMPONENT (IV)</th>
<th>COMPONENT (V)</th>
<th>COMPONENT (VI)</th>
</tr>
</thead>
</table>
| Software requirements specification for each module  
• Test cases tracker  
• User manual for each module | Module definition  
• Modules category definition  
• Inputs/outputs location | Set of macros  
• Testing convention of macros  
• Macros, inputs & outputs | Select modules  
• Modifying parameters of macros, inputs & outputs | High-level description of result  
• Detailed results  
• Well-structured output layout | Assist the applicability of checks (includes metadata and tools) (high level standard structure) |

Development Testing Release Support

Considerations

Though the overall experience was positive and helped with identifying issues, the user requirements phase turned out to be the most problematic. This reaffirmed that a "Data Scientist" needs to have really broad knowledge.