Applying ISO/IEC 25010 Standard to prioritize and solve quality issues of automatic ETL processes

Tiago L. Alves*, Pedro Silva*, and Miguel Sales Dias*†
*Microsoft Language Development Center, Lisbon, Portugal
†ISCTE - University Institute of Lisbon (ISCTE-IUL), Lisbon, Portugal
Emails: { t-tialve, v-pedros, midias }@microsoft.com

***Abstract*—The goal of any automatic process is to run without human intervention. However, it is not uncommon that unexpected issues recur, requiring repetitive human intervention in order to processes execute successfully. Such situations indicate that those processes were not designed with the necessary quality requirements to achieve the automation goal, and quality needs to be improved.**

This paper describes a case study of the quality improvements of several automatic ETL processes that required a growing amount of effort to ensure its successful execution. Key to the improvement process was the use of the ISO/IEC 25010 Standard for Systems and Software Quality Requirements and Evaluation to categorize all issues found. This approach proved beneficial in two important ways: first, it created a clear understanding of the overall ETL processes quality problems; second, it made obvious which issues deserved uttermost attention.

I. Introduction

Developing a tool that automates a daily critical process requires understanding the individual process' tasks as well as the exceptional conditions that must be handled. When such tools are in production it is expected that few unexpected conditions arise requiring human intervention.

In an ideal scenario those tools provide good monitoring and reporting mechanisms to warn when unexpected situations must be manually handled. Furthermore, when such situations are found, the tool is continuously evolved to automatically handle them ensuring future minimum intervention. It is more common, however, that execution of the daily critical processes require a large amount of (manual) effort, typically spent on understanding what caused the tool to fail (or, in some extreme cases, understanding if the tool succeeded at all) and handling unexpected situations. In a dantesque scenario this effort is increasing over time frequently causing the inability to guarantee that the processes run within their available time window.

This paper describes the application of the ISO/IEC 25010 Standard for Systems and Software Quality Requirements and Evaluation [1] to improve the quality of several critical daily automatic extract, transfer, and load (ETL) processes at Microsoft. Each known quality issue was categorized with one of the standard's quality Characteristics and Sub-characteristic, and with a label defining how each issue would be handled. From this categorization process it became very easy to communicate the issues to the management, pinpoint the priority ones and create an action plan to solve them. After solving the top priority issues, the automatic ETL processes were able to execute with no intervention for several weeks.

This paper is structured as follows. Section II presents a brief overview about the ETL automatic processes and the problem definition. Section III introduces the ISO/IEC 25010 Standard. Section IV describes the methodology followed to categorize the ETL's quality issues using the standard's quality characteristics and sub-characteristics, plus an additional categorization to identify the actions to be taken for each issue. The results and their applicability are presented in Section V, related work in Section VI, and the conclusion in Section VII.

II. Problem definition

Several ETL processes were developed to supply daily data for an internal web-portal. Each ETL process handles a specific data domain and is made available via a web-portal. Both the nature of data and the web-portal purpose are confidential.

In general, each ETL process follows the *extract*, *transform*, and *load* steps as defined by Kimball [2]. However, the implementation of these steps is specificagiven the nature and volume of data, and the technology involved. Going into detail, each ETL performs the following identical steps: *(i)* A batch-job is submitted from the server to a parallel-processing cloud-based environment. The job is responsible for extracting data from one or many text-based files, perform the necessary transformations, and create a processed data file; *(ii)* The processed data file is downloaded to the server and bulk loaded to a database; *(iii)* Several database and filesystem tasks are executed to remove unneeded data and temporary files.

As mentioned, these ETL processes were requiring a growing amount of effort to guarantee its successful execution. There were a multitude of reasons causing the ETL processes to fail, e.g. unavailability of the extracted files at the batch-job execution, temporary connectivity problems, and processed files attempted to be downloaded before being produced. In addition, each ETL implementation was done independently having no shared components. Daily execution logs recorded thousands of entries having frequently several megabytes of size. Finally, the original development team of these processes was no longer available and no documentation existed.

While it was important to handle each issue individually (the ETL processes must continue to run), it was equally important to lay a strategy on how to solve these issues. The main challenge was to define which issues are causing more impact and consequently had to be solved first. To accomplish this we used the ISO/IEC 25010 Standard as a framework to categorize all the issues found, use these categorizations to create a broader overview of all problems and identify the priority issues that had to be solved first.

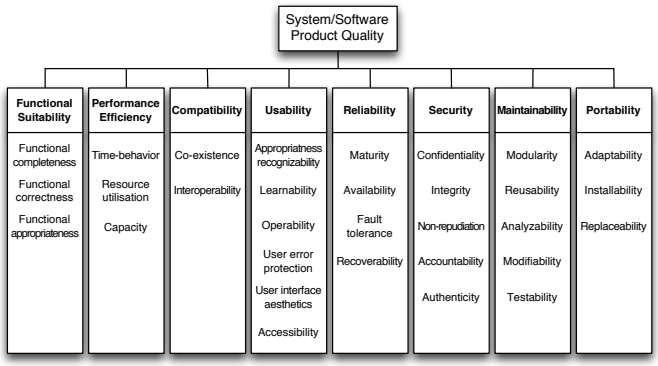

Fig. 1. ISO/IEC 25010 Standard Quality Characteristics and Sub-characteristics as defined in [1].

III. ISO/IEC 25010:2011 STANDARD

The ISO/IEC 25010:2011 Standard for Systems and Software Quality Requirements and Evaluation [1] defines a framework to specify and evaluate software product quality. This framework describes eight main characteristics divided into several sub-characteristics (Figure 1). For brevity we will define the quality characteristics and enumerate the sub-characteristics.

Functional Suitability, defined as "degree to which the software product or system provides functions that meet stated and implied needs when used under specified conditions", is sub-divided into: *functional completeness*, *functional correctness*, and *functional appropriateness*.

Performance Efficiency, defined as "performance relative to the amount of resources used under stated conditions", is sub-divided into: *time behavior*, *resource utilization*, and *capacity*.

Compatibility, defined as "degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware and software environment", is sub-divided into: *co-existence*, and *interoperability*.

Usability, defined as "degree to which a product or system can be used by specified users to achieve goals with effectiveness, efficiency, and satisfaction in a specified context of use", is sub-divided into: *appropriateness recognizability*, *learnability*, *operability*, *user error protection*, *user interface aesthetics*, and *acessibility*.

Reliability, defined as "degree to which a system, product or component performs specified functions under specified conditions for a specified period of time", is sub-divided into: *maturity*, *availability*, *fault tolerance*, software faults"; and *recoverability*.

Security, defined as "degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization", is sub-divided into: *confidentiality*, *integrity*, computer programs or data" *non-repudiation*, repudiated later" *accountability*, and *authenticity*.

Maintainability, defined as "degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers", is sub-divided into: *modularity*, *reusability*, *analysability*, *modifiability*, and *testability*.

Portability, defined as "degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another", is sub-divided into: *adaptability*, *installability*, and *replaceability*.

These characteristics and sub-characteristics enable abstraction over a broad range of quality issues. This is important in two complementary ways. First, by deciding upon the quality characteristics that are more critical to the product/system success it becomes easy to focus on a set of related issues to solve. Second, because this categorization is comprehensive, it decreases the risk of overlooking (possibly) important issues due to their dependencies.

IV. METHODOLOGY

The methodology had three steps: *Categorization and grouping*, *Action labeling* and *Dependency analysis*.

A prerequisite is having a list of identified issues. Each issue was described using a single-sentence. The strict rule of a single sentence description had a twofold purpose: reduce the problem scope to the essential (hence easing the categorization) and limit the size of overall recorded issues by preventing duplicates. An example of an issue was "Database bulk-load failure due to change of downloaded file columns and/or column sizes".

In the first methodology's step, *Categorization and grouping*, each issue was categorized with a single quality sub-characteristic and grouped under the same quality-characteristic. The above issue, for instance, was categorized under the *Fault tolerance* quality sub-characteristic, defined in the standard as "degree to which a system, product or component operates as intended despite the presence of hardware or software faults", and grouped under the *Reliability* quality characteristic. Cases where multiple sub-characteristics were applicable provided indication that more than one issue was involved. These general issues were split into several more specific ones and categorized individually. Doing so, although not required, helped to understand better the issue's root-cause and simplified the decision on how to handle that issue. There were no cases where an issue could not be split or where it was advantageous to use multiple sub-characteristics. Also, no issues were found that could not be categorized using a standard's quality sub-characteristic.

In the *Action labeling* step, each issue was labeled with a specific keyword indicating the action to take in order to fix it. Four different labels were used: *Improve*, *Report*, *Monitor*, and *Document*. The *Improve* label was used to indicate a process' change to automatically handle the issue. This label had the foremost importance. An example of such issue was "Time-consuming and execution errors due to manual deployment of ETL processes", categorized under the *Installability* sub-characteristic which is defined in the standard as "degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified

environment". The *Report* label was used to indicate a change in the way issues were reported. This label was used when the ETL process normal execution stopped and human intervention was required. In such cases, it is desirable that the ETL process reports the exact issue cause and (if possible) provide guidelines on how to solve it. An example of such issue was given earlier at the beginning of the section. Another issue was "Failure to download log file due to temporary unavailability of disk space in temporary storage", categorized under the *Resource utilisation* quality sub-characteristic which is defined in the standard as "degree to which the amounts and types of resources used by a product or system, when performing its functions, meets requirements". The *Monitor* label was used to indicate a change in the monitoring capabilities that were used to provide an overview of the ETL process status and to predict potential issues. An example of such issue was "Large number of messages when pooling for the processed log availability", categorized under the *Operability* sub-characteristic which is defined in the standard as "degree to which a product or system has attributes that make it easy to operate and control". Two more issues were "Track the evolution of daily records ingested in the database" and "Track the database growth evolution", categorized under the *Capacity* sub-characteristic defined in the standard as "degree to which the maximum limits of product or system parameter meet requirements". Finally, the *Document* label was used to indicate the need for documentation the issue's resolution. There was a single issue using this label, "Difficulty in finding the necessary binary and source code files of the ETL processes", categorized under the *Learnability* sub-characteristic which is defined in the standard as "degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use".

In the *Dependency analysis* step, focus was put on the identification of issues that could hinder the general resolution of others (e.g. infrastructural issues, productivity/performance bottlenecks). This step ensured not to overlook lower priority issues that could seriously impact the resolution of the top priority ones. An example of such issue was already mentioned before in the *Action labeling step* for the *Improve* label (lack of automatic means to deploy the ETL processes). Another example of such issue, categorized under the *Reusability* sub-characteristic, was "Common ETL process parts must be changed individually due to lack of component sharing". These two issues fall in the *Portability* and *Maintainability* quality characteristics respectively, and, in general, were not so relevant when compared, for instance, with *Reliability*. However, because a series of changes was being planned their relevance greatly increased. We additionally identified low-effort issues that optionally could be easily solved together with the more critical ones. Doing so opened the possibility to improve the overall product/system quality.

The methodology's first step, *Categorization and group* was responsible for creating a clear high-level overview about the product/system specific quality issues. Most important issues were found in the characteristics most relevant to the product/system in question. The second step, *Action labeling* had a twofold responsibility of validating the issues' categorization by giving a direction towards the solution and creating awareness about the effort that would be necessary to fix them.

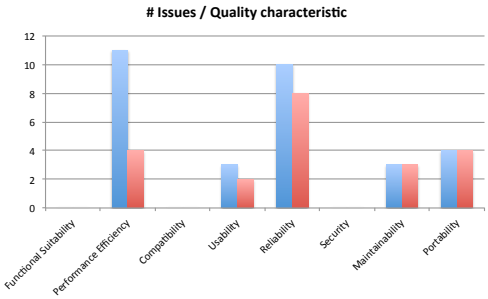

Fig. 2. Number of unique issues per quality characteristic defined in the ISO 25010 Standard. The first column represents all issues, while the second column represents only the issues labeled as *Improve*.

Finally, the third step, *Dependency analysis* ensured that issues that had impact on fixing others more important were not overlooked. After this final step, it was clear the most important issues that had be solved first.

Note that this methodology followed a bottom-up approach starting by categorizing all known issues and only later focusing on the most important quality characteristics. In contrast, a top-down approach would start by defining the quality characteristics that are most important to the system/product, and then focusing only on the issues that fall under those characteristics. The bottom-up approach was chosen deliberately to address the lack of previous knowledge about the ETL processes and their issues. Neither the original team was available nor documentation existed. By listing all known issues it was ensured that no specific quality characteristics were overlooked. This was important, for instance, to identify the *Maintainability* and *Portability* issues and to recognize that solving them first would ease the process of solving the issues falling under other quality characteristics.

V. Results and Applicability

The top priority of the ETL processes was to guarantee their successful execution. However, a growing amount of effort was needed to ensure that, on a daily-basis, each ETL process executed. Given that the original development team was no longer available, a new team took the ownership of those processes. The new team had no previous experience with such processes. In the beginning, a brief description of each issue was written on a daily basis, with the initial purpose of creating a walk-through document on how to handle them. In the first four weeks of doing so it became clear that many of those issues were frequently repeating. The methodology described in this paper evolved, in first place, from the need to create a shared overview of the overall quality problems. By grouping and categorizing the issues in the standard's characteristics and sub-characteristics it was easy to communicate to management that there was a problem that needed a solution.

Following the described methodology over 30 unique issues for all ETL processes were identified. Because each issue was trimmed down to its root-cause and handled multiple times, it was straightforward to identify the actions to take in order to handle them. Considering issues labeled *Improve*

as the most critical[1] ones, the total number of issues was reduced to 21 (over 30% decrease). Figure 2 depicts all issues (first column) and issues labeled as *Improve* (second column) per quality characteristic. For the *Performance efficiency* quality characteristic, 9 issues belonged to *Resource utilisation* and 2 to *Time-behavior* sub-characteristics. For the *Usability* quality characteristic, 2 issues belonged to *Operability* and 1 to *Learnability* sub-characteristics. For the *Reliability* quality characteristic, 7 issues belonged to *Fault tolerance*, 2 to the *Recoverability* and 1 to the *Availability* sub-characteristics. For the *Maintainability* quality characteristic, all 3 issues belonged to the *Reusability* sub-characteristics. Finally, for the *Portability* quality characteristics, 2 issues belonged to *Adaptability* and 2 to *Installability* sub-characteristics.

The majority of issues was expected to be found under the *Reliability* quality characteristic. However, given the existent *Maintainability* and *Portability* issues, each *Reliability* change would have to be done in each individual ETL process (i.e. the cost of solving each *Reliability* issue would be multiplied by the number of ETL processes). This supported the decision to solve the *Maintainability* and *Portability* issues first, and then the *Reliability* issues all at once. The communication with management was straightforward. The problem was clearly demonstrated, a thorough investigation of all issues was done, the most important ones (which were aligned with the processes' priorities) were listed, and a justified plan with the issues to be solved was presented.

The action plan defined solving the *Maintainability* and *Portability* issues first, as they were fundamental for saving effort on future changes. Then, issues were solved for the *Reliability*, *Performance efficiency* and *Usability* quality characteristics, by this order of priority, as these characteristics were most important for the ETL processes. From over 30 issues identified, 15 were solved (50% of the overall identified) requiring an effort of three man-weeks. After these changes, all ETL processes ran for several weeks without reporting a single intervention. In contrast, before those changes, the daily cost of ensuring all processes executed successfully amounted to one man-day.

The methodology introduced in this paper was applied to several identical ETL processes with a manageable number of issues (50). Two main limitations were identified when generalizing this work to arbitrary systems. First, a different set of *Action* labels might be necessary. The *Report* and *Monitor* labels might not apply to other systems, and the *Improve* label might be too generic. However, in principle, the concept of *Action* labels should equally apply in helping the prioritization process taking into consideration that they are meant to provide a direction on the solution that should be adopted for each issue, a means of creating smaller clusters issues and, that it should be clear that each cluster has different weight in terms of impact to the overall system. Second, larger systems might have many more issues which might be harder or too time-consuming to categorize. In principle, such large systems can be decomposed into smaller ones, making the whole process more tractable. Additionally, this methodology does not require a previous exhaustive analysis of all issues. An iterative application of this methodology would work by analyzing of a manageable number of issues (e.g. 50-100), prioritizing them, solving them, re-evaluating the overall improvements, and then re-iterating. Finally, we should take into account the frequency of each issue in order to aid the prioritization process and to define a cut-off line for issues to be solved.

[1]Issues labeled with *Improve* have the bigger impact on the processes, because they deal with situations that could be automatically handled.

VI. Related work

Our work is related to Chillarege *et al.* [5] by categorizing defects and inferring from those categorizations where to focus to prevent those defects. However, while our goal was create an quality overview over all issues and prioritize those to be solved first, Chillarege's goal was to pinpoint the software development phases that require changes. Other related work found follow different approaches to quality, namely modeling and measurement. Pavlov [3], as example of the first case, introduces a quality-of-experience model, based on ISO/IEC 25010 Standard, which focuses on the relevance of different quality characteristics to architectural parts of an ETL system. Bakota *et al.* [4], on the other hand, follow an approach of deriving an assessment of the *Maintainability* quality characteristic based on source code metrics. Our work, in contrast, makes use of the ISO/IEC 25010 Standard as a framework to aid the prioritization of quality issues that need to be solved.

VII. Conclusion

In this paper we introduced a methodology to prioritize the resolution of a system/product quality issues. This methodology follows a bottom-up approach, categorizing known issues using the ISO/IEC 25010 Standard quality characteristics and sub-characteristics, labeling each issue with an *Action* indicating a direction of solution, and then finding the dependencies between them. The overall process enabled the identification of the key issues that needed to be solved and assisted in the development of an action plan. We successfully applied this methodology to evolve a set of ETL processes at Microsoft for which neither documentation existed nor original team was available. The methodology proved beneficial in creating a shared clear overview of the existent quality issues and helping to determine which issues should be handled first.

References

[1] ISO/IEC, "ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models," International Organization for Standardization, Tech. Rep., 2010.

[2] R. Kimball and J. Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. John Wiley & Sons, 2004.

[3] I. Pavlov, "A QoX model for ETL subsystems: Theoretical and industry perspectives," in *Proc. of the 14th Int. Conf. on Computer Systems and Technologies*. ACM, 2013.

[4] T. Bakota, P. Hegedüs, I. Siket, G. Ladányi, and R. Ferenc, "Qualitygate SourceAudit: A tool for assessing the technical quality of software," in *CSMR-WCRE*. IEEE, 2014.

[5] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M.-Y. Wong, "Orthogonal defect classification-a concept for in-process measurements," *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 943–956, Nov. 1992.