# Towards the automation of a defect detection protocol for functional size measurements

Denisse Madrigal-Sánchez, Christian Quesada-López, Marcelo Jenkins

University of Costa Rica, San Pedro, Costa Rica
{denisse.madrigal, cristian.quesadalopez, marcelo.jenkins}@ucr.ac.cr

**Abstract.** There is a need to develop formal protocols to verify the accuracy of software functional size measurements and to calibrate the measurement processes. These protocols offer a mechanism for accomplishing such verification regardless of who performed the measurements (an expert or an automated tool). The detailed analysis of functional size measurement procedures results is necessary to understand the nature of the defects. The automation of these protocols decreases the effort and time required to verify the measurements accuracy and supports the calibration of procedures and tools for functional size measurement. This paper presents an empirical study for evaluating the results of a prototype tool that automates an accuracy verification protocol for IFPUG FPA measurements. Our tool uses a graph-based model to search for the causes of measurement mismatches detected in the verification process. The current prototype has reached an accuracy of 93.92%, a precision of 98.69% and a recall of 94.94%.

**Keywords:** Functional size measurement, defect detection tool, accuracy verification protocol automation, IFPUG FPA, empirical study

## 1 Introduction

Functional size measurement is one of the main factors influencing effort and cost estimations in a software development project [1]. There are five standardized methods for functional size measurement: COSMIC-FFP (ISO/IEC 19761), IFPUG FPA (ISO/IEC 20926), MkII (ISO/IEC 20968), NESMA (ISO/ IEC 24570) y FiSMA (ISO/IEC 29881). Historically, IFPUG FPA is the most popular one [2] and COSMIC-FFP is the one with the most increased adoption in the last years [3]. Both methods measure functionality delivered to final user by doing function points analysis.

Research related to procedures and tools for automated functional size measurement has increased in the last years [4]. With such increase, the need for research of empirical studies to verify measurement results obtained through these automated procedures has also been identified [4,5]. Symons [6] establishes that there are many functional size measurement procedures available in literature that should be carefully used by professionals. The author explains that an automated procedure is reliable when it has been calibrated and validated previously in the context of application measurement.

M. Genero, M. Kalinowski (Eds.): CIbSE 2018, Bogotá - Colombia, 2018

It is necessary to define mechanisms to verify the accuracy and reliability of results obtained by measurement procedures. Soubra, Abran & Ramdane-Cherif [4] state the need to define and use standardized accuracy verification protocols in order to proof the accuracy of functional size measurement procedures independently.

An accuracy verification protocol for functional size aims to evaluate the accuracy of functional size measurements by comparing control values (obtained by an expert) to results obtained by third parties as professionals applying measurement procedures or tools that automate such measurement. Accuracy verification protocols have been proposed and validated for measurement results following COSMIC-FFP and IFPUG FPA standards [4,7,5].

Manual execution of verification protocols is error prone, time and effort consuming. The prototype tool offers a mechanism to automate the execution of an accuracy verification protocol saving time and effort. The prototype tool could also provide support during the calibration of procedures and the identification and analysis of assignable causes for measurement errors.

We present a prototype tool that automates an accuracy verification protocol for functional size measurements based on method IFPUG FPA. We also present the results of an empirical evaluation to validate to prototype tool effectiveness. In the context of this study, our prototype reached an accuracy of 93.92%, a precision of 98.69% and a recall of 94.94%.

This paper is structured as follows: Section 2 presents the related work on accuracy verification protocols for functional size measurements. Section 3 describes the prototype tool implemented. Section 4 describes the empirical study conducted to validate the prototype tool effectiveness. Section 5 shows the results of the tool evaluation. Finally, Section 6 outlines conclusions and future work.

## 2 Related work

Protocols for accuracy verification report measurement results obtained through an specific procedure and compare them to the measurement results obtained by an expert [4,5]. In [8], a set of verification protocols were identified, in the following we briefly describe these protocols.

Soubra, Abran & Ramdane-Cherif [4] developed a protocol that allows the verification of accuracy in all steps of the measurement procedure of an automated tool implementing the standard COSMIC-ISO 19761. The protocol requires measurement results to be compared to measurement results manually obtained by a human expert so that the automated tool can be calibrated. Verification in every step allows to keep traceability between software input artifacts and functional measurement components. Bagriyanik & Karahoca [7] used Soubra et al. protocol [4] to verify the accuracy of measurements generated by an automated tool for COSMIC-FFP measurements. Authors keep the original 3 phases but also add two more steps for detailed verification in phase 2.

Yilmaz, Tunalilar & Demirors [9] developed a tool for automatic defect detection in COSMIC-FFP measurements. Authors also presented a methodology

to validate the efficiency of their tool in terms of correctness and accuracy. Functional users trigger events to show function points and objects of interest verification. The tool takes such verification to conduct the analysis and generate a report of detected defects. In this study, error categories are associated to three error causes: Measurer Related, Measurement Process Related and Software Requirements Specification documented related. In order to evaluate the tool efficiency this study also required manual defect detection that was then compared to the results obtained by the tool.

The studies we presented so far refer to protocols for COSMIC-FFP measurements which is not the measurement method of our case of study. Nevertheless Soubra, Abran & Ramdane-Cherif and Bagriyanik & Karahoca studies provide the baseline for the accuracy verification protocol automated by our prototype tool. Quesada-López & Jenkins [5] adapted Soubra et al. protocol [4] to verify the accuracy of IFPUG FPA measurements. Measurements results are verified using a top-down evaluation to compare total unadjusted function points(UFP) and basic functional components (BFC). For every BFC, this protocol analyzes total number of data element types (DET), record element types(RET) and file types referenced (FTR). Such granular analysis allows to detect errors in the calculation of complexity of the BFCs. Same as [4], authors highlight the importance of keeping traceability between requirements and input artifacts and measurement results so that the protocol not only identifies measurement differences but their root cause.

Finally, Morris & Desharnais [10] described a verification method based on historic IFPUG FPA measurement data. The validation identifies errors based on variances to expected norms. Deviation from the norm happens when measurement results do not comply with measurement results specific for the context of the application. Based on historic data and application specific characteristics, ranges of variation for a specific BFC are determined. Measurements verification based on historic data and variances to expected norm is an approach different to the one proposed by Quesada-López & Jenkins so it can not be used as support for our prototype tool.

## 3 Prototype tool

Quesada-López & Jenkins [5] propose an accuracy verification that uses a top down evaluation of functional size measurement results in order to identify differences reported by applying one or more measurement processes. The protocol looks for consistency comparing measurement results differences against a true value reported by an expert. We selected this protocol as the one automated by our prototype tool because this protocol is designed to work with IFPUG FPA measurements and this standard is the subject of study of our research. Quesada-López & Jenkins [11] conducted a controlled experiment to compare functional size measurements for two different measurement procedures: IFPUG FPA and Automated Function Point (AFP). Two subsequent replications of this experiment were then conducted in [12] and they were analyzed using the verification protocol proposed by Quesada-López & Jenkins [5].

### 3.1 Protocol phases

The protocol has 3 phases: 1. Total UFP measurement results comparison 2. Detailed comparison of the accuracy of measurement results. 3. Identification and errors recovery. We analyzed the protocol phases in order to determine whether or not the level of detail of those was enough to accomplish the protocol automation and to determine which steps of the protocol procedure would required improvements. Here we discuss improvements applied:

- Delayed execution of phase 3: The protocol executes phase 3 whenever an error is identified in phase 2 so that errors in inputs (or procedures) are fixed and the measurement results are re-evaluated to verify the calibration success.The prototype delays the execution of phase 3 until a full scan of all measurement results is performed. Such delay allows to identify the status of all measurements before trying to find assignable causes for errors.
- Functional elements comparison: According to the protocol, measurement results have to be sorted by requirement, function and type. From automation point of view, it is necessary to come up with a mechanism to guarantee that the exact same elements are compared. We defined these requirements to perform numerical comparisons of measurement results: (1) Elements containing numerical attributes need an identification property. (2) Such property must be consistent between the *true value*[1] measurements and the subject[2] measurements.

To solve later requirements, we defined *pair measurement*. A measurement is a pair measurement of another one if they both have the exact same name. The prototype will only compare numerical data if it could previously pair two measurements.

**Protocol coverage and prototype model** The prototype implements phases 1 and 2 of the protocol. Phase 3 requires human intervention to inspect the quality of requirements, input artifacts and measurement procedure. The automation of this kind of inspection is out of the scope of this study, but a proposal to look for assignable causes based on the prototype data model and the measurement errors identified in phase 2 is described in Section 3. For phase 3 the prototype identifies and reports the measurement differences.

- Measurement results are represented with a graph data structure. Each set of measurements (subject and true value) are represented on its own graph.
- The prototype uses deep first algorithm to traverse the graphs and visit all nodes in both graphs simultaneously. In each node, measurements are compared and evaluated.

[1] *True value* measurement results refer to measurement results obtained by an expert and that will be considered true value when evaluating measurement results through the protocol

[2] Subject measurements results refer to measurement results that need to be evaluated, either obtained by a human being or a tool

- The graphs traversal to evaluate measurements and the search for assignable causes generate the results that are presented to the user.

The prototype was implemented using Java as programming language and *Spring-boot* framework. We selected *Neo4j*as graph database and we also used *Spring Data Neo4j* library for graphs processing.

**Prototype process model** The protocol execution requires input data loaded into the prototype. Such input data includes measurement results to be evaluated and measurement results that will be considered as true value. Once input data is loaded, it needs to be mapped to the prototype data model and then the prototype can start executing the protocol. Measurement differences and measurement errors are reported to the user and they are also used as input for the similarity analysis proposed in order to search assignable causes.

### 3.2 Measurement results modeling

**Model design** We used a graph to represent all measurements in a count of the same application performed by a single source, either the expert or the person or tool providing the measurement results that will be evaluated. The graph nodes represent characteristics of the measurement of a functional element: The count it belongs to, requirement, function type and basic functional components associated, data logical groups and data element types. Edges represent relations between these characteristics.

**Model implementation** Based on the design from previous section we implemented the model in Figure 1. Graph nodes are represented by a base class named Node that contains all attributes of a graph node. All other node types extend from Node. CountRoot node contains the number of data functions and transactional functions associated to a count. DFRoot node allows to group all data function requirements which are at the same time composed by data functions. TFRequirement nodes have the name of the artifact that presents the requirement. TransactionalFunction nodes show the type of the transactional function, the UFP size and the number of DETs and FTRs associated to the function. TransactionalFunction is similar to DataFunction, but the last one has different types and a attribute for RETs instead of FTRs. Graph edges allow us to keep traceability between a measurement and the elements it is associated to. More details of the prototype tool implementation can be found in [13].

### 3.3 Procedure modeling

**Model design** We defined some concepts as part of the procedure design phase. These need to be understood for a better understanding of the prototype procedure model:

- Subject node: Node currently analyzed in the subject graph.
- True value node: Node in the true value graph that corresponds to the subject node.

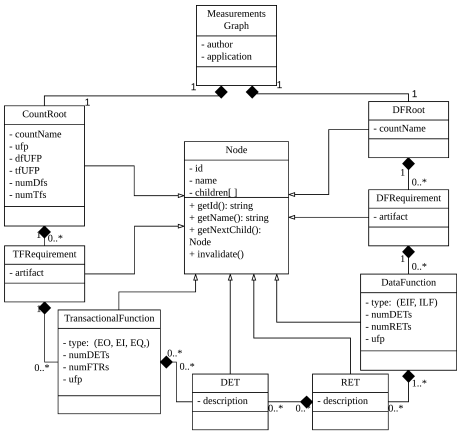


Fig. 1: Implemented Data Model

- Subject graph: Measurement results graph currently being evaluated.
- True value graph: True value measurement results.
- Pair node: Node that matches the name and path of a node in the counterpart graph.
- Invalidated node: Node that has no pair node in the counterpart graph.

As mentioned earlier, the prototype procedure takes advantage of deep first graph traversal. Such algorithm adapts naturally to the problem so that one measurement at a time is analyzed prioritizing nodes in deeper levels. Figure 2 shows the process flow followed by the prototype to implement protocol phases. Each sub-process tagged represents an algorithm implemented in the prototype.

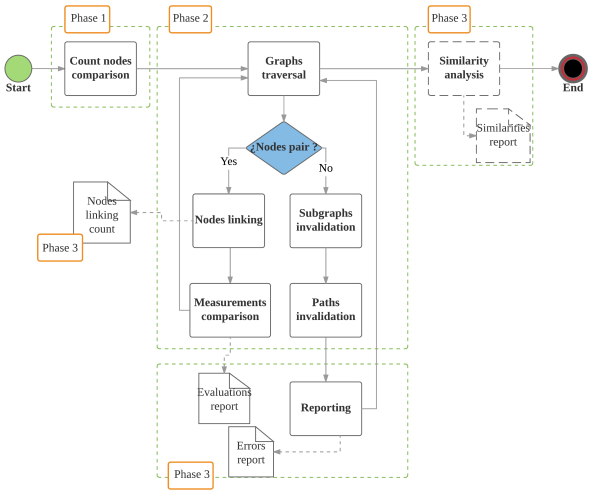


Fig. 2: Prototype execution process of protocol phases

The prototype first compares count nodes an register the evaluation results. Then the graphs traversal begins. Such traversal looks for pair nodes in order

to allow the visit of a deeper level node. If the pairing between subject and true value nodes succeed, nodes are linked and the pairing is registered, then pair nodes measurements are evaluated. If the pairing fails, the algorithm has detected an invalidated path. This causes invalided nodes to be tagged and reported. This process continues until the entire top sub-graph is traversed. After this, a second traverse starts in the bottom sub-graph from the DF node. The proposal for sub-graphs similarities would be applied once both sub-graphs are traversed in both subject and true value graphs.

**Model implementation** The model design includes eight algorithms described now:

- Count nodes comparison: This algorithm implements phase 1 of the protocol by comparing total size measurement results. Total unadjusted function points and the total number of DFs and TFs are compared.
- Graphs traversal: This algorithm traverses both graphs using deep first algorithm. Detailed measurement results are compared if pairing succeeds. Otherwise, differences are reported.
- Nodes pair: This algorithm allows to determine if two given nodes, one from subject graph and another from true value graph, actually pair.
- Sub-graphs invalidation: This algorithm is triggered when it is possible to continue traversing a graph but its counterpart graph can no longer be traversed because a node was invalided or the total number of nodes between both graphs does not match. The algorithm invalidates an entire sub-graph with a root node that was just invalidated.
- Path invalidation: This algorithm is similar to the previous one but its goal is to invalidate a path. This algorithm is invoked by the previous one.
- Reporting: This algorithm is invoked every time a node is invalidated. It allows to keep track of the specific paths that generated an invalidated node.
- Node linking: This algorithm marks two nodes, one from subject graph and another from true value graph, as pairs.
- Measurements comparison: This algorithm generates metrics about the evaluation of two specific measurements, one from subject graph and another from true value graph. It uses magnitude of relative error (MRE), magnitude of error relative (MER) and balanced relative error (BRE).

**Proposal for assignable causes search based on invalidated sub-graphs** The execution of algorithms previously described allows to gather information about invalidated paths and paired nodes. When the execution of all these algorithms finishes, the search for assignable causes takes place. Such search has two characteristics:

- Search based on invalidated sub-graphs must be bi-directional. This means to look for subject invalidated sub-graphs in true value graph and to look for true value invalidated sub-graphs in subject graph.

- Sub-graphs search must be a similarity search. This means that if exact sub-graphs cannot be found in counterpart graph, then such sub-graph needs to be adjusted to run new searches. Adjustments need to continue until proofing no sub-graphs of the initial sub-graph are found in the counterpart graph.

The process to drive the similarity search based on invalidated sub-graphs is described now. Given a invalidated sub-graph:

- If the root is a requirement node, search through all requirement nodes in the counterpart graph, one with the same structure of BFC.
- If the root is a transactional function type (EI, EO, EQ) or a data function type (EIF, ILF), search through all function types in the requirement being analyzed in the counterpart graph, one with the same structure. If there is no match continue searching through remaining requirements.
- If the root is a RET node, search through all RET nodes in the requirement being analyzed in the counterpart graph, one with the same structure. If there is no match continue searching through remaining requirements.
- If the root is a DET node, this will be considered a lost node.

Previous process will report all similarities identified and will support the measurements analyst on the process of errors detection by reducing the amount of time and effort required to execute a verification protocol.

## 4 Tool evaluation

In order to evaluate the prototype tool we compared the measurement results obtained manually by one of the researchers using the protocol against the results obtained automatically by the tool. The evaluation reports the tool effectiveness to detect measurement errors. Based on the prototype results, we also reported results of measurement defect findings for measurement procedures applied to a small transactional application by two researchers with measurement expertise (true value measurement results) and a random selection of 7 measurement counts performed by 12 software engineering professionals. We selected 7 cases because it is the minimum number of measurements evaluated for one group in tests ran by [9]. Finally we analyzed the most common errors and we suggest assignable causes for those.

### 4.1 Case study design

This study aimed to answer two research questions:

1. Is the prototype tool effective to find measurement differences, report errors and suggest assignable causes? This research question allows to determine if the tool implements the protocol correctly and the tool effectiveness to detect defects and suggest assignable causes.
2. What are the assignable causes identified by the tool based on the measurement results evaluation? This research question allows to identify the most common type of errors and assignable causes identified by the tool for functional size measurement procedures.

We collected two groups of metrics in order two answer each research question. The first group represents metrics related to functional size measurements accuracy that have been used in previous studies [1,14,11]. These metrics allow the evaluation of measurements accuracy in regards to true value measurements: We collected MRE, MER and BRE to measure accuracy. We also collected total function points and size measurement for BFC components. To measure reproducibility we collected absolute value of difference between subject count and average count for the same sample. The second group of metrics allows to evaluate the tool effectiveness to detect defects based on the comparison against control values. We used confusion matrixes as described in [15,16]. We collected confusion matrix, accuracy, precision and recall.

### 4.2 Case study implementation

These steps describe how we implemented the case study:

1. Selection of software application to use as object of study and the elaboration of requirements specification following standard IEEE-830 [17]. The selected application has 144 UFP and the requirements document and the source code of the application are very similar to examples in a real management information system (MIS) in the industry.
2. Functional size measurement of the selected application. 2 researches with measurement experience measure the application. Results are considered true value.
3. Twelve professionals collect functional size measurement for the selected application using the IFPUG FPA method. The subjects were mainly developers and testers of a graduate metrics course, not the usual users of FSM measurement methods but familiar with software engineering practices. In order to accomplish leveling, first participants assist to four training sessions. Next, they measure the application using requirements specification as baseline and reporting all measurement results. After that, measurement results are reviewed to validate they are detailed enough to work as prototype tool inputs. Finally, seven cases are selected randomly to evaluate the prototype.
4. Accuracy verification protocol is applied manually to identify all differences and assignable causes between the true value measurements and the 7 cases selected. This step is executed by one of the researches and results are considered control values.
5. Prototype is designed and implemented to automate the accuracy verification protocol.
6. Measurement results are processed to generate inputs required by the prototype.
7. Verification protocol is applied with the prototype to identify measurement defects and assignable causes for such differences.
8. Prototype reports findings and accuracy verification results.
9. Prototype's results are compared against the control values using metrics mentioned early in this section.

### 4.3 Threats to validity

*Internal Validity:* Human factor could influence the results of manual procedures. Measurement differences between participants was reduced by selecting subjects with similar experience in the application of functional size measurement (FSM). Experiment artifacts could also influence results. The same requirements specification was used by all participants. The reduced number of participants in the sample also represents a thread, whereas the fact of all being industry practitioners is an advantage for this study. Prototype inputs were prepared in a semi-manual process that also represents a thread in the correctness of the Neo4j scripts. Control values used to evaluate the prototype results and the participants measurements were generated by one researcher who applied the accuracy verification protocol. This could influence the control values. Lastly, two researchers of this study, who have measurement experience, applied the FPA method to do the functional size measurement. They do not have a CFPS certification. *External Validity* Although the application analyzed is a small one (144 UFP), its requirements are similar to industry real cases. Participants were mainly analysts, developers or testers. It means they are not the ones who usually do functional size measurement but they do have understanding about software engineering practices. A similar performance is expected between all participants. *Construct Validity* Measurements reliability is an important consideration for the validity of conclusions about results. All measurements used in this study for the functional size evaluation and accuracy are based on known FSM methods proven in literature and standard ISO 14143-3 [18]. Confusion matrixes are also proven in literature as a way to evaluate results quality in a classification problem.

## 5 Results

Tool effectiveness was evaluated based on the results it generated when comparing participants results to true value results. Table 1 shows the results of measurements evaluation obtained by the prototype tool for total functional size and data functions (DFs) and transactional functions (TFs). Median Magnitude of error relative (MdMER), Median Magnitude of relative error (MdMRE) and Median Balanced relative error (MdBRE) are acceptable for TFs considering a variance of 10%. Standard deviation for such measures is around 10%. Results for DFs show MdMRE and MdBRE of 75% whereas MdMER is 42.86%. This means subjects' total size measurements for DF components are different to true value measurements. Standard deviation for DF components total size is 13.57%, this means results are closer to mean than TF components results even for the total size reported. Total size measurements show better results than DF components results. Total size results reproducibility is similar to total size results for TF components. DF components show reproducibility results almost 30% close to the mean value.

Table 2 shows measurement evaluation results for EI, EO and EQ components. EI and EQ show acceptable results with mean and median values for MRE, MER and BRE lower than 10%. EO components show results slightly higher

Table 1: Measurement evaluation results

| | TF | | | | | | DF | | | | | | Total | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UFP | MRE | MER | BRE | Diff Rep | | UFP | MRE | MER | BRE | Diff Rep | | UFP | MRE | MER | BRE | Diff Rep | |
| **Mean** | 110.4 | 11.7% | 13.2% | 13.7% | 13.6 | 14.5% | 49.1 | 75.5% | 38.5% | 75.5% | 21.1 | 28.3% | 159.6 | 17.2% | 15.7% | 18.1% | 24.7 | 14.5% |
| **Median** | 110.0 | 10.3% | 11.5% | 11.5% | 12.0 | 14.2% | 49.0 | 75.0% | 42.9% | 75.0% | 21.0 | 18.1% | 161.0 | 18.8% | 15.8% | 18.8% | 27.0 | 8.2% |
| **Std Dv** | 17.2 | 9.3% | 12.3% | 12.5% | 10.8 | 13.2% | 13.6 | 48.5% | 19.9% | 48.5% | 13.6 | 34.4% | 24.6 | 9.3% | 8.8% | 10.1% | 13.3 | 18.9% |
| **Min** | 84.0 | 1.7% | 1.8% | 1.8% | 2.0 | 1.9% | 28.0 | 0.0% | 0.0% | 0.0% | 0.0 | 0.9% | 112.0 | 2.8% | 2.7% | 2.8% | 4.0 | 1.6% |
| **Max** | 139.0 | 27.6% | 38.1% | 38.1% | 32.0 | 41.4% | 70.0 | 150.0% | 60.0% | 150.0% | 42.0 | 98.2% | 188.0 | 30.6% | 28.6% | 30.6% | 44.0 | 55.6% |

Table 2: Measurement results for EI, EO and EQ

| | EI | | | | | | EO | | | | | | EQ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UFP | MRE | MER | BRE | Diff Rep | | UFP | MRE | MER | BRE | Diff Rep | | UFP | MRE | MER | BRE | Diff Rep | |
| **Mean** | 58.7 | 7.4% | 7.9% | 8.2% | 4.4 | 8.8% | 18.0 | 15.8% | 18.1% | 18.6% | 3.0 | 0.0% | 15.8 | 5.5% | 6.2% | 6.2% | 1.0 | 9.8% |
| **Median** | 61.0 | 4.8% | 5.0% | 5.0% | 3.0 | 2.9% | 18.0 | 15.8% | 18.1% | 18.6% | 3.0 | 20.0% | 17.0 | 3.9% | 3.7% | 3.9% | 1.0 | 0.0% |
| **Std Dv** | 6.2 | 7.0% | 7.8% | 8.0% | 4.1 | 12.0% | 3.5 | 6.1% | 10.0% | 9.3% | 1.2 | 24.0% | 10.3 | 6.1% | 7.0% | 7.0% | 1.0 | 16.2% |
| **Min** | 48.0 | 0.0% | 0.0% | 0.0% | 0.0 | 0.4% | 15.0 | 10.5% | 9.5% | 10.5% | 2.0 | 0.0% | 0.0 | 0.0% | 0.0% | 0.0% | 0.0 | 0.0% |
| **Max** | 64.0 | 17.2% | 20.8% | 20.8% | 10.0 | 30.7% | 21.0 | 21.1% | 26.7% | 26.7% | 4.0 | 40.0% | 27.0 | 13.3% | 15.4% | 15.4% | 2.0 | 38.9% |

than 10%. This means participants' results were close to true value results. A low UFP standard deviation for the three components also proofs mentioned closeness. Reproducibility has better results in EI and EO components.

None of the participants reported EIF components and all reported 28 UFP of ILF components. This matches with true value results. ILF components then have perfect accuracy and reproducibility for all participants.

Results show that the prototype can evaluate measurements automating the generation of accuracy metrics measurements for each evaluation executed by the prototype. Results also match with those shown in [12], according to the last one, for the group of 12 subjects, accuracy showed better results for TF components than DF components.

### 5.1 Tool effectiveness for error reporting and assignable causes suggestion

Measurement errors detected by the prototype were compared against control values obtained by one of the researchers. According to Figure 3, the prototype tool detected 901 errors correctly, it also reported 12 false positive cases. After analyzing this cases we determined those cases are nodes the prototype could not pair because of differences upper/lower case differences between subject and true value node name. The prototype tool reported 48 false negative cases. False negative cases are due to one of two reasons:

- Subject reported duplicated measurements.
- Subject reported measurements incompletely.

| | | Prediction P | Prediction N | | |
|---|---|---|---|---|---|
| Real | P | 901 | 48 | **Accuracy** | 93.92% |
| | N | 12 | 26 | **Precision** | 98.69% |
| | | | | **Recall** | 94.94% |

Fig. 3: Tool effectiveness for error detection

These scenarios cannot be loaded into the graph database, the first one because nodes cannot have duplicated names and the second one because it represents missing information. Since data cannot be loaded into the database, it cannot be analyzed by the prototype tool. The prototype tool reported 26 true negative cases for nodes that matched the name and tag correctly. Based on

Table 3: Tool effectiveness for error detection by error category

| Error Categories | Accuracy | Precision | Recall | Error Categories | Accuracy | Precision | Recall |
|---|---|---|---|---|---|---|---|
| **Incorrect Classification** | **100.0%** | **100.0%** | **100.0%** | **Unnecessary Report** | **100.0%** | **100.0%** | **100.0%** |
| *Incorrect Association* | 100.0% | 100.0% | 100.0% | *Unnecessary BFC* | 100.0% | 100.0% | 100.0% |
| *Incorrect Type* | 100.0% | 100.0% | 100.0% | *Duplicated DET* | 100.0% | 100.0% | 100.0% |
| **Tagging Error** | **96.4%** | **96.1%** | **100.0%** | *Unnecessary BFC* | 100.0% | 100.0% | 100.0% |
| *DET Tag does not match* | 96.4% | 96.1% | 100.0% | *Duplicated Logical Group* | 100.0% | 100.0% | 100.0% |
| **Missing Report** | **100.0%** | **100.0%** | **100.0%** | *Duplicated Requirement* | 100.0% | 100.0% | 100.0% |
| *Missing DET* | 100.0% | 100.0% | 100.0% | *Unnecessary Requirement* | 100.0% | 100.0% | 100.0% |
| *Missing Message* | 100.0% | 100.0% | 100.0% | | | | |
| *Missing Requirement* | 100.0% | 100.0% | 100.0% | | | | |

Table 4: Error categories and types not detected by the tool

| **Missing Report** | **14** | **Unnecessary Report** | **34** |
|---|---|---|---|
| *Missing Association* | 14 | *Duplicated Association* | 34 |

previous results and for our case study scenario, the prototype tool detects defects correctly 93.92% of times. When the prototype tool detects a defect, it does it correctly 98.69% of times and when there are errors, the tool detects them 94.94% of times. Receiver operating characteristic curve (ROC) analysis to evaluate the tool effectiveness to detect defects show ROC equals to 0.82. This means the tool is classified as good on defect detection.

Besides reporting invalidated paths, the tool provides error categories. Such categories where defined by one researcher based on a proof of concept executed to validate the prototype tool. Table 3 shows prototype tool effectiveness for defect detection based on the error category. The prototype tool detects all errors categorized as incorrectly classified, unnecessary reports and missing reports. For tagging errors category, the tool can detect defects with 96.37% accuracy.

Regarding the errors not detected by the tool, Table 4 shows the number of cases present in the measurements analysis done by one researcher. As explained earlier, these cases correspond to scenarios in which the database does not have all the information, either because it is missing or because it is already reported.

### 5.2 Assignable causes identified by the tool based on the measurement results evaluation

These are the assignable causes identified by the tool after processing invalidated paths:

- Incorrect classification: The user identified a measurement correctly but classifies it incorrectly. Incorrect association refers the RETs reported by the user in incorrect BFCs. Incorrect type refers to BFCs reported with incorrect type.
- Tagging errors: Refers to DETs with names that do not match.
- Missing report: Refers to cases in which subject does not report a measurement but it does show in the true value measurement results. Missing DETs, missing button or messages reports and missing requirements.
- Unnecessary report: Refers to subjects reporting elements that do not show in the true value graph. BFCs reported by the subject but not by the true value, duplicated DETs with the same name but different tags, unnecessary DETs reported in BFCs, duplicated RETs, requirements with duplicated names but different tags and requirements reported by the subject but not by the true value.

### 5.3 Preliminary results for the similarity search

In order to validate the proposal for similarity search we conducted a proof of concept (PoC) based on the algorithm proposed for the similarity search. Our proof of concept is based on a subset of true value measurement results for Contoso application and a subject measurement created from seeded errors in the true value measurement. The prototype would report all invalidated paths and the sub-graphs with some similarity in the counterpart graph. The PoC showed the report of all invalidated paths for subject measurements. In this case, 13 paths were invalidated. 3 of them did not get a suggestion for similar sub-graphs.Two paths referred to paths that do not exist in the true value graph because they represent unnecessary reports. One path represented a data element reported by the true value but not by the subject.

## 6 Conclusions and future work

This study presents a prototype tool that automates an accuracy verification protocol for IFPUG FPA functional size measurements. The tool generates the report of detected defects and suggests assignable causes for such defects by categorizing them. To validate the tool we elaborated a case study to compare accuracy verification results obtained automatically by the prototype against control values obtained by three researchers manually. Such validation includes reporting the prototype tool effectiveness to find defects previously identified by researchers. Evaluation results show the effectiveness of the prototype tool to report measurement defects and suggest assignable causes for such defects. Results analysis determines that the prototype tool can detect four measurement error categories: Incorrect classification, tagging error, missing report and unnecessary report. These categories are associated to possible assignable causes by the error classification in each category. As future work, the accuracy verification protocol and the prototype tool can be used to verify functional size measurement of any procedure following a model (or a subset) with the same basic functional components of IFPUG FPA. A component to register measurement results in the prototype tool would not only reduce the time and effort required to load measurements results in the tool. It would allow to remove the risk of having a human being translating measurements to scripts that can then be imported into the database. Similarly the tool could be extended so that measurement results validation is also reported automatically by the tool. In regards to errors categorization, tags evaluation mechanism needs to be improved so that node names and tags are not an exact text match but a text similarity comparison. The component for assignable causes suggestions can be extended to include more scenarios. This require the execution of more cases and the evaluation of different algorithms for sub-graphs similarity detection.

## References


1. S. M. Abrahao, "On the functional size measurement of object-oriented conceptual schemas: Design and evaluation issues," Ph.D. dissertation, 2004, aAI3154436.
2. S. Fingerman, *Practical software project estimation: a toolkit for estimating software development effort duration.* Boston, MA: McGraw Hill, 2011.

3. A. Abran and R. Dumke, “Cosmic function points: Theory and advanced practices,” 2011.
4. H. Soubra, A. Abran, and A. Ramdane-Cherif, “Verifying the accuracy of automation tools for the measurement of software with cosmic – iso 19761 including an autosar-based example and a case study,” in *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, Oct 2014, pp. 23–31.
5. C. Quesada-López and M. Jenkins, “Applying a verification protocol to evaluate the accuracy of functional size measurement procedures: An empirical approach,” in *Proceedings of the 16th International Conference on Product-Focused Software Process Improvement*, New York, 2015.
6. C. Symons, “Lies, damned lies and software metrics,” in *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, Oct 2014, pp. 174–175.
7. S. Bagriyanik and A. Karahoca, “Automated cosmic function point measurement using a requirements engineering ontology,” *Information and Software Technology*, vol. 72, no. Supplement C, pp. 189 – 203, 2016.
8. C. Quesada-López and M. Jenkins, “Procedimientos de medicin del tamao funcional: un mapeo sistemtico de literatura,” ser. CIBSE 2017, 2017.
9. G. Yilmaz, S. Tunalilar, and O. Demirors, “Towards the development of a defect detection tool for cosmic functional size measurement,” in *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, Oct 2013, pp. 9–16.
10. P. Morris and J. Desharnais, “Function points analysis. validating the result.” ser. TotalMetrics, 2001.
11. C. Quesada-López and M. Jenkins, “An evaluation of functional size measurement methods,” in *Ibero-American Conference on Software Engineering*, Lima, Peru, 2015.
12. C. Quesada-López, D. Madrigal, and M. Jenkins, “An empirical evaluation of automated function points,” in *Ibero-American Conference on Software Engineering*, Curran Associates, Inc. Quito,Ecuador: Curran Associates, Inc., 2016.
13. D. Madrigal, “Automatizacion de un protocolo de verificacion de la medicion del tamano funcional,” Universidad de Costa Rica, San Pedro, Costa Rica, Tech. Rep., 2017. [Online]. Available: https://goo.gl/aAfy7f
14. S. Abrahao, G. Poels, and O. Pastor, “Evaluating a functional size measurement method for web applications: an empirical analysis,” in *10th International Symposium on Software Metrics, 2004. Proceedings.*, Sept 2004, pp. 358–369.
15. N. V. Chawla, *Data Mining for Imbalanced Datasets: An Overview.* Boston, MA: Springer US, 2005, pp. 853–867.
16. in *Data Mining (Fourth Edition)*, fourth edition ed., I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Eds. Morgan Kaufmann, 2017, pp. i – iii.
17. IEEE, “Ieee recommended practice for software requirements specifications,” *IEEE Std 830-1998*, pp. 1–40, Oct 1998.
18. ISO/IEC, “Information technology - software measurement - functional size measurement. part 3: Verification of functional size measurement methods,” International Organization for Standardization, Geneva, CH, Standard, 2007.