Original Manuscript ID: Access-2025-05493

Original Article Title: "Refining Software Clustering: The Impact of Code Co-Changes on Architectural Reconstruction"

To: IEEE Access Editor

Re: Response to reviewers

Dear Editor,

Thank you for allowing a resubmission of our manuscript, with an opportunity to address the reviewers' comments.

We are uploading (a) our point-by-point response to the comments (below) (response to reviewers, under "Author's Response Files"), (b) an updated manuscript with yellow highlighting indicating changes (as "Highlighted PDF"), and (c) a clean updated manuscript without highlights ("Main Manuscript").

Best regards,

Adelina Stana and Ioana Sora

Reviewer#1, Concern # 1 (Still, the language can be improved in terms of readability (the grammar looks pretty OK), limitations could be discussed more extensively, the discussion on practical implications also could be deepen.)

Author response: Thank you for the suggestion. We tried to improve readability by refactoring parts of the paper (Abstract, Introduction, Structural and Logical Dependencies, Experimental results).

Reviewer#2, Concern # 1 (Is it really clear the term "Software clustering for architectural reconstruction"?):

Author response: We understand that "software clustering for architectural reconstruction" may need clarification. In our context, software clustering refers to the process of grouping related software entities (such as classes or files) based on various types of dependencies to recover the system's modular structure. Architectural reconstruction refers to the activity of reverse engineering a system's architecture from its implementation artifacts.

Author action: We updated the abstract and introduction to clarify what we meant by "software clustering for architectural reconstruction."

Reviewer#2, Concern # 2 (Revise phrasing, e.g. "Improve the understanding" could be "Improve understanding")

Author response: Thank you for the suggestion.

Author action: We updated the abstract and removed the phrasing "improve the understanding".

Reviewer#2, Concern # 3 (There is literature that treats the logical and co-change dependencies as synonymous, while others refine them based on statistical significance or frequency. This should be clearer.)

Author response: In our work, we follow the second perspective: logical dependencies are co-changes that remain after refinement using different filters.

Author action: We updated the abstract to clarify that logical dependencies in our work are refined from co-change data using different filters.

Reviewer#2, Concern # 4 ("Evaluate their performance using two clustering evaluation metrics." which ones?)

Author response: The Modularization Quality (MQ) metric and the MoJoFM metric.

Author action: We updated the abstract to explicitly name the two evaluation metrics used in the paper.

Reviewer#2, Concern # 5 (What are the limitations in extracting structural dependencies?)

Author response: We are referring to cases such as dynamically typed languages like Python or JavaScript, where static code analysis is unreliable due to the lack of explicit type information. In such contexts, structural relationships are difficult to extract.

Author action: We added a paragraph related to the limitations of structural dependencies in Section III.A.

Reviewer#2, Concern # 6 (What are "good results"?)

Author response: We agree the phrase was vague. We meant that logical dependencies produced results comparable to those of structural dependencies based on MQ and MoJoFM scores.

Author action: We removed the phrase "good results" from the abstract to avoid unintended conclusions.

Reviewer#2, Concern # 7 (The text suggests that combining logical and structural dependencies improves clustering results. If the study is evaluating whether improvement occurs, it should be phrased as a hypothesis rather than a conclusion.)

Author response: Our intention was to briefly highlight the results.

Author action: We updated the abstract to formulate the potential improvement from combining logical and structural dependencies as a hypothesis rather than a conclusion.

Reviewer#2, Concern # 8 (The text says "software systems often need more documentation". Does it mean lack documentation or need additional documentation?. In general, revise phrasing.)

Author response: We intended to say that software systems often lack sufficient or up-to-date documentation.

Author action: We rephrased the sentence.

Reviewer#2, Concern # 9 ("Code co-changes/logical dependencies (relationships between entities extracted from the version control system)": suggests that logical dependencies and co-change dependencies are the same, but this is not necessarily true. Logical dependencies can be filtered co-changes, but there are different interpretations of logical dependencies in the literature.)

Author response: We agree that, as currently phrased, the sentence may suggest that logical and co-change dependencies are the same. As stated in the abstract and throughout the paper, we consider logical dependencies to be filtered co-changes. However, this particular sentence does not reflect that distinction.

Author action: We rephrased the sentence to better distinguish logical dependencies from raw co-change dependencies.

Reviewer#2, Concern # 10 ("Does using structural dependencies (SD) combined with logical dependencies (LD) improve software clustering results compared to traditional approaches using only structural dependencies (SD)?" - all traditional clustering approaches do really only rely on SD?)

Author response: We agree that not all traditional clustering approaches rely solely on structural dependencies.

Some prior work incorporates other dependencies, including lexical or co-change information. However, many widely used approaches and tools (e.g., Bunch, ACDC, ARTs) rely primarily on structural dependencies.

Author action: We updated the text to clarify that structural dependencies are commonly used in traditional clustering approaches but not exclusively.

Reviewer#2, Concern # 11 (The text lacks details on the specific filtering criteria. Does "rare co-changes" mean low-frequency changes, and how is "true dependency" defined? The filtering strategy should be explicitly justified.)

Author response: Indeed, the term “rare co-changes” needs clarification. In our work, this refers to low-frequency co-changes. “True dependencies” are co-changes that occur more frequently and are considered more reliable.

Author action: We revised the introduction and briefly described the filters applied to co-changes.

Reviewer#2, Concern # 12 ("We then evaluate the results using two metrics: MQ (Modularization Quality) and MoJoFM (Move and Join Effectiveness Measure)." why these metrics were chosen?)

Author response: MQ and MoJoFM were chosen because they provide complementary views: MQ assesses internal structure, while MoJoFM compares against a reference architecture. Each helps address limitations that may arise when using the other alone.

Author action: We updated the introduction to briefly explain why MQ and MoJoFM were used and how they complement each other.

Reviewer#2, Concern # 13 (The main contributions are missing.)

Author response: Indeed, the contributions were not explicitly listed.

Author action: We addressed this by adding a summary of the main contributions at the end of the introduction.

Reviewer#2, Concern # 14 (The need / innovation / the problem to be solved is missing.)

Author response: We agree that the motivation and problem statement needed to be more clearly stated.

Author action: We revised the introduction to better explain the problem addressed by this work.

Reviewer#2, Concern # 15 (The introduction mentions related work only in Section II, but it should at least provide a brief contextualization of existing approaches within the introduction; which is key.)

Author action: We updated the introduction to include related work to support the context of our research.

Reviewer#2, Concern # 16 (Improve the flow and length.)

Author action: We considered this feedback while addressing the previous comments related to the introduction.

Reviewer#2, Concern # 17 (The main goal of this section is not only list works, it must also explain what are the main similarities and differences between existing works and this proposal.)

Author action: We revised the related work section and added the similarities and differences.

Reviewer#2, Concern # 18 (Are all structural dependencies equally reliable? Cases where they might fail?)

Author response: Structural dependencies are extracted using static code analysis; their influence varies, which is why we assign weights based on dependency types (e.g., interface realization is more impactful than local variable usage). If we refer to reliability in the same context as co-changes reliability, then for our experiments, structural dependencies are all considered reliable, and we do not question whether a structural dependency actually exists in the code. Failures may occur, especially with low-weight structural dependencies like local variables, when they are only instantiated but never used. We do not address this particular case regarding structural dependency reliability because code quality tools are usually applied to software systems, and developers are prevented from integrating code that contains issues such as 'unused local variable'.

Author action: We added a paragraph related to the limitations of structural dependencies in Section III.A.

Reviewer#2, Concern # 19 (Why is the commit size threshold set to 20?)

Author response: We set the commit size threshold to 20 based on observations from our previous studies, where we analyzed commit distributions across multiple systems and found that commits with more than 20 files often included unrelated changes that introduced noise into logical dependencies and significantly increased their volume. Other authors have also set commit size thresholds relatively close to ours: Capiluppi et al. considered commits with fewer than 10 modified source code files; Ying et al. used a threshold of fewer than 100 files; Zimmermann et al. configured the ROSE tool to exclude commits with more than 30 files; and Moonen et al. recommended a threshold of 8 files.

Based on our previous studies, a threshold under 10 cuts out a large number of commits from the commit history, and it is not uncommon for developers to modify 10 files in a single update. On the other hand, setting the threshold to 100 would mean that a single commit with 100 changed files introduces 4950 co-changes, which in some systems could result in up to 8 times more co-changes than the number of structural dependencies extracted from the system.

Author action: We updated Section III.B.1 with a brief clarification to better explain why the commit size threshold is set to 20.

Reviewer#2, Concern # 20 (The confidence formula does not account for projects with varying commit sizes.)

Author response: That is correct; the confidence formula is applied to co-changes extracted from commits with the commit size filter applied.

Author action: We interpreted this remark as a statement rather than a request for clarification or change, so we

did not modify the manuscript. Section III, Figure 1 illustrates and explains the filtering process.

Reviewer#2, Concern # 21 (Why were these specific weights chosen?)

Author response: We interpreted this question as referring to the weights assigned to logical dependencies. In our approach, logical dependency weights represent the number of times two entities were changed together in the commit history. This reflects the frequency of co-change, which serves as an indicator of their logical connection strength. We wanted to have similar behavior for LD as for SD, where more significant structural dependencies are assigned higher weights than others (for example, interface realization over local variable usage). Since the average weights assigned to logical dependencies are close in value to the average weights assigned to structural dependencies, we kept this method for assigning weights.

Our intention was to ensure that the logical dependency weights align with the structural dependency scale so that structural dependency influence in the dependency graph is not diminished.

Author action: We updated Section III.B.4 with a clarification regarding the weights assigned to logical dependencies.

Reviewer#2, Concern # 22 (Why combine SD and LD instead of treating them separately?)

Author response: We combine SD and LD because they capture complementary aspects of software relationships. SD represents explicit code-level connections, while LD reflects links based on change history. In our experiments (Sections V and VI), we also treat them separately to evaluate their impact, but we also analyze them together in order to assess which type of dependency performs better: SD, LD, or the combination of SD and LD

Author action: We have reformulated texts in different parts of the paper, and we hope that it is more clear now that we have investigated all 3 cases: LD and SD separated and combined.

Reviewer#2, Concern # 23 (Why these clustering algorithms were chosen?)

Author response: We selected Louvain, Leiden, and DBSCAN because they are scalable to systems of different sizes and compatible with the structure of our input data. Another important criterion was that the clustering algorithms should not require the number of clusters to be specified in advance. This is important in our context, where the number of expected modules is unknown. Louvain and Leiden are community detection algorithms designed for graph-based data, aligning with how we represent structural and logical dependencies. Leiden was included alongside Louvain because it addresses some of Louvain's limitations.

DBSCAN was chosen because it offers a different clustering perspective. It can discover clusters of varying shapes and identify noise in the data. Using these three algorithms allows us to observe better how the type of input dependencies influences clustering results.

Author action: We updated Section IV.A to explain why Louvain, Leiden, and DBSCAN were selected.

Reviewer#2, Concern # 24 (The text does not specify if the logical dependencies are extracted using a standard method, tool, or heuristic.)

Author response: Logical dependencies are extracted using a custom tool developed in our previous work. The tool description and workflow is described in Section III.B.4 of the paper.

Author action: We updated the first sentence of Section III.B.4 to clarify that logical dependencies are extracted using a tool developed in our previous work.

Reviewer#2, Concern # 25 (There is no mention of whether dependency weights are normalized when combining multiple files.)

Author response: In our approach, dependency weights are not normalized when combining structural and logical dependencies. Instead, we sum the weights directly when the same pair of entities appears in both types. The weight values reflect the relative importance of each dependency type.

Author action: We updated Section III.B.5 to clarify that when combining structural and logical dependencies, the weights are summed without normalization.

Reviewer#2, Concern # 26 (Please explain figure 3.)

Author response: Figure 3 is explained in the processing step of Section IV.C (Workflow for Software Clustering and Evaluation). It illustrates the internal steps of the tool: reading dependencies, building the graph, applying clustering algorithms, and computing evaluation metrics.

Author action: We updated the figure caption and moved the explanation of the figure to the beginning of the subsection, right before the "Input" paragraph.

Reviewer#2, Concern # 27 (why MoJoFM was chosen?)

Author response: We chose the MoJoFM metric because it gives an external point of view on the clustering solution by comparing it to a manually defined reference architecture. This complements the MQ metric, which reflects the internal structure. Compared to the original MoJo metric, MoJoFM normalizes the result to a percentage scale (0–100%), making it easier to interpret and compare across different systems. Also, MQ can sometimes favor over-segmentation by rewarding small, connected clusters. Using MoJoFM alongside MQ helps identify such cases (we have not encountered any in our experiments so far).

Author action: We updated Section IV.B to clarify that MoJoFM offers an external perspective on clustering, complementing MQ's internal view. We also included the motivation for choosing MoJoFM over MoJo.

Reviewer#2, Concern # 28 (How were the reference solutions created, and why is manual inspection considered reliable?)

Author response: Manual inspection is considered reliable in this context because it involves analyzing the system's structure from a human perspective, taking into account naming conventions, package organization, class responsibilities, and available documentation. We do not consider the manually created reference solution to be an absolute ground truth; we use it as a practical baseline. A significant difference between the reference solution and the clustering result would raise a flag, indicating potential issues in the clustering result. Indeed, the human perspective can be subjective; this is why we also rely on the MQ metric, which is computed automatically and does not depend on human input.

Author action: We updated Section IV.B.2 to clarify how the reference solutions were constructed.

Reviewer#2, Concern # 29 (Are weights capped or normalized? How does weight accumulation affect clustering quality?)

Author response: Weights are not capped or normalized in our approach. When combining structural and logical dependencies, weights are accumulated by summing the individual weights from each type of dependency for the same entity pair. This accumulation increases the influence of that entity pair in the clustering process. The effects of this accumulation on clustering results can be observed in Tables 4 to 7 in the rows labeled SD+LD.

Author action: No further update was made, the normalization of weights was clarified in Section III.B.5 in response to an earlier related comment from the reviewer.

Reviewer#2, Concern # 30 (Why was CSV chosen instead of formats like JSON or database storage?)

Author response: We chose CSV because it is straightforward to represent the dependency data as a flat structure of entity pairs with associated weights. It is also easy to generate, read, and process using standard tools and libraries, which made it a practical choice for our experimental workflow. Since the data does not require nested structure or complex querying, more advanced formats like JSON or database storage were unnecessary.

Author action: No paper update was made; the choice of CSV format was a technical implementation detail that did not directly impact the experimental methodology or results.

Reviewer#2, Concern # 31 (The text mentions that the tool constructs a directed graph but does not mention how self-loops, duplicate edges, or cycles are handled.)

Author response: Self-loops cannot occur in our implementation because all input data is represented as pairs of distinct entities. A self-loop would imply that an entity depends on itself, which is not a situation reported by our dependency extractor tools. If duplicate edges between the same pair of entities appear in multiple input files, they are handled by summing their weights. Cycles are not explicitly removed, as the clustering algorithms we use (Louvain, Leiden, and DBSCAN) operate on the full graph structure and are not affected by the presence of cycles.

Author action: We updated Section V.B (Processing) to clarify that self-loops are absent due to input format constraints and explain how duplicate edges and cycles are handled.

Reviewer#2, Concern # 32 (If DBSCAN is applied to a directed graph, is it adapted for that structure?)

Author response: In our implementation, DBSCAN is not directly applied to the directed graph. Instead, we transform the dependency graph into a distance matrix by interpreting the edge weights symmetrically, combining the weights in both directions (e.g., from A to B and from B to A). This transformation ensures the distance matrix is symmetric, making it suitable for DBSCAN.

Author action: We updated Section V.B (Processing) to clarify that the directed graph is converted into a distance matrix before applying DBSCAN, making it compatible with the algorithm requirements.

Reviewer#2, Concern # 33 (MQ can vary across implementations, how is it computed here?)

Author response: The MQ metric is computed as defined in formula [3] from Section IV.B.1 (MQ Metric).

Author action: No paper update was made regarding this remark, the MQ formula used in our implementation is included in Section IV.B.1.

Reviewer#2, Concern # 34 (The text says "all commits" were used for logical dependencies, but for structural dependencies, only the code at the "specific tag" was used. why are structural dependencies only extracted from one version while logical dependencies are extracted across multiple commits?)

Author response: Structural dependencies are extracted from a single version (at a specific tag) because they represent the system's static architecture at a fixed point in time. Logical dependencies are mined from the project's evolution history, which requires analyzing co-changes across multiple commits. Extracting structural dependencies across all commits would be redundant and computationally expensive.

Author action: No paper update was made regarding this remark.

Reviewer#2, Concern # 35 (There are threshold values (10 to 100 in steps of 10), but why start at 10? Why is 100 the maximum? How does changing the threshold affect clustering?)

Author response: The explanation for why 100% is the maximum threshold is provided in the paper, in Section VI.B. It represents the strictest filtering possible, as the strength metric is clipped to a maximum value of 100. The threshold values are applied to the strength metric, which calculates the percentage of commits involving entity A that also involve entity B, scaled by a system factor. Because the strength metric is derived from actual co-change frequency, a strength value of 0% is impossible. There must be at least one co-change event between two entities to compute a valid strength value. We start at 10% as a practical lower bound to avoid including a large number of low-confidence, noisy dependencies.

Changing the threshold affects the number and quality of logical dependencies included in clustering. Tables 4–7 show how different threshold values impact MQ and MoJoFM scores across systems.

Author action: We updated Section VI.B to better clarify these aspects.

Reviewer#2, Concern # 36 (Why are ten runs performed for logical dependencies but only one for structural dependencies?)

Author response: Ten runs are performed for logical dependencies because each run corresponds to a different strength threshold value applied during co-change filtering. We generate a new logical dependency file for each threshold.

Author action: We updated Section VI.B. to clarify that each logical dependency strength threshold results in a separate input file and, therefore, a separate clustering run.

Reviewer#2, Concern # 37 (There is no mention of the computing environment.)

Author response: All experiments were run on a machine with an Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz, 2.11GHz. However, the focus of our study is on the quality of clustering results rather than execution time or performance, so the specific hardware does not impact the conclusions.

Author action: We added a new subsection under the Experimental Plan (Computing Environment) to describe the computing environment for running the experiments.

Reviewer#2, Concern # 38 (Were experiments run on a local machine or a cloud-based environment?)

Author response: All experiments were run on a local machine.

Author action: We added a subsection under the Experimental Plan (Computing Environment) to clarify that all experiments were conducted locally.

Reviewer#2, Concern # 39 (The claim that SD+LD covers 100% of the system and therefore is the best approach is a bit dangerous. Coverage does not necessarily imply better clustering quality. Logical dependencies may contain important relationships that structural dependencies miss, meaning that blindly prioritizing coverage could lead to lower-quality clusters.)

Author response: We fully agree that coverage alone does not guarantee better clustering quality, and we did not intend to claim that SD+LD is the best approach solely due to its ability to cover 100% of the system. Our choice is based on practical use: It is generally preferable to have clustering results that include most or all entities in the system in real-world clustering scenarios. LD-only, especially at 100%, may have high MQ or MoJoFM scores but sometimes covers less than 10% of the system's entities, which limits its practical utility. Having 10% of the systems classes assigned with high precision into modules while knowing nothing about the other 90% of classes will be of no practical use in many reverse engineering activities.

Our preference for SD+LD is based on the fact that it offers broader system coverage while producing better or comparable clustering quality in many cases. Additionally, by combining structural and logical dependencies, we do not lose the co-change relationships that SD alone might miss. Therefore, the combination keeps the valuable connections introduced by LD.

Author action: We updated the conclusion section to clarify that we do not blindly prioritize coverage when evaluating clustering quality. Instead, we highlight that the combination of SD and LD improves system coverage and retains the important co-change relationships that SD alone may miss.

Reviewer#2, Concern # 40 (The text states that LD-only clustering provides the best quality but does not cover the entire system. However, then it suggests that SD+LD is the best approach due to full system coverage.)

Author response: This remark overlaps with the previous remark regarding the relationship between coverage and clustering quality.

Author action: We hope that the updates already mentioned also cover this.

Reviewer#2, Concern # 41 (the 10-40% as the best range for SD+LD and 100% for LD alone do not have theoretical or statistical reasoning.)

Author response: That is correct; these observations are empirical and based on the experimental results we obtained across the evaluated systems. We do not claim theoretical or statistically validated generalizations for these thresholds. The identified ranges (e.g., 10–40% for SD+LD and 100% for LD) emerged from patterns observed during experimentation, and we report them as such.

Author action: The updates in the paper and the newly introduced paragraph on cross-systems experiments hopefully help support our empirical observations.

Reviewer#2, Concern # 42 (The assumption that higher MQ implies better clustering ignores the possibility of over-segmentation.)

Author response: We agree that a higher MQ value may favor over-segmentation by increasing intra-cluster cohesion. For this reason, we complement MQ with the MoJoFM metric, which evaluates clustering quality against a reference solution and helps detect over-segmentation. Together, these metrics provide a more balanced assessment of clustering quality.

Author action: We updated the end of the MQ subsection (Section IV.B.1) to mention the potential limitation of MQ.

Reviewer#2, Concern # 43 (Are differences in MQ and MoJoFM values between SD, LD, and SD+LD statistically significant, or are they within an error margin?)

Author response: To address this, we conducted a statistical analysis using the Friedman test. The results showed that the differences between the three input configurations (SD-only, SD+LD(20), and LD(100)) are statistically significant in terms of MQ for all clustering algorithms. For MoJoFM, the differences were also statistically significant for Louvain and Leiden but not for DBSCAN.

Author action: We added a new subsection **"Cross-System Results Comparison"** in the paper (Section VII.E).

Reviewer#2, Concern # 44 (How is the strength threshold of 10%, 20%, etc., calculated? Is it based on absolute values, percentile distributions, or another method?)

Author response: The strength threshold is a filtering value applied to the percentage-based strength metric scores. For example, a threshold of 10% filters out all logical dependencies whose strength metric is below 10%. The strength metric calculation is described in Section III.B.2. It is based on a confidence-like measure that calculates the percentage of commits involving entity A that also involve entity B, representing the strength of their co-change relationship. To avoid favoring entity pairs that co-occur in a small number of commits (which may inflate the percentage), the result is then scaled by a system factor, which adjusts for commit activity across the entire project. Since this scaling can push the value above 100%, we clip the final strength score to remain within a valid percentage range of 0 to 100. The strength threshold is expressed in percentages because the strength metric represents a percentage.

Author action: We updated Section III.B.2 to specify that strength metric values are all expressed as percentages to avoid confusion about how the strength threshold is applied.

Reviewer#2, Concern # 45 (Does a high MQ value in Apache Ant correspond to the same clustering quality as in Tomcat?)

Author response: MQ values are normalized between 0 and 1, which makes them comparable across systems.

Author action: No paper update was made regarding this remark.

Reviewer#2, Concern # 46 (How dependencies were extracted from the codebase. Were they derived from static analysis, runtime analysis, or another method?)

Author response: Structural dependencies were extracted using static code analysis, based on the inspection of source code relationships.

Author action: We updated Section V. to clarify that structural dependencies are extracted using static code analysis.

Reviewer#2, Concern # 47 (Were experiments run on the same machine? How were results averaged across multiple runs?)

Author response: Yes, all experiments were run on the same local machine. For scenarios involving multiple runs (e.g., varying the logical dependency strength threshold), results were computed independently for each run and reported per threshold. No averaging was applied across runs; each run represents a distinct configuration with its input data.

Author action: No update was made for this remark; the computing environment and execution setup were already addressed in response to a previous remark.

Reviewer#2, Concern # 48 (Add the exact version of Apache Ant, Tomcat, etc.)

Author response: The exact versions of all systems used in our study, including Apache Ant and Tomcat, are already listed in Table 3 "Overview of projects used in experimental analysis."

Author action: We updated the header of the "Release Tag" column in Table 3 to "Release Tag (Version)" to indicate that it refers to the version used in the experiments.

Reviewer#2, Concern # 49 (General: the document needs clarity, order, justification and identity. Also, the background should be background, the method should be method. Later on, please update the references to fully support your work.)

Author action: We refactored big parts of the paper and hope that this brings more clarity.

Reviewer#3, Concern # 1 (Fine for publication.)

Thank you for the appreciation of our work.

Reviewer#4, Concern # 1 (If you give a formula for confidence, you should also give a formula for support.)

Author response: Thank you for the remark. We agree that including the formula for support adds clarity to the explanation of the strength metric.

Author action: We added the formula for the support metric alongside the confidence formula in Section III.B.2.

Reviewer#4, Concern # 2 (Figure 4 - in its current form does not contribute much. It would be useful to add in the middle rectangle an example of the tool that had the greatest impact in a given scenario.)

Author response: The figure illustrates the different scenarios we use in our experiments with our own tool, which is responsible for both clustering and evaluation. Since only a single tool is involved, it is not applicable to specify one as having greater impact over others.

Author action: No changes were made to Figure 4, even though we agree that its explanation is already included in the text. The figure visually confirms and summarizes what is already described in the paper.
