Specific comments:

1. *The paper presents new tool for interactive natural hazard risk analysis evaluation. The solution is not novel, it is based on existing, open-source geospatial software architecture.... However, I would appreciate more detail on choices of the system architecture. Authors used a package of tools offered by Boundless and as they do not evaluate the selection of tools, it seems they used it because it was available in a package. I'm not suggesting that authors need to re-write or extend the paper, but in a paper about a web-GIS architecture used in such a specific context as natural hazard risk analysis, I would prefer seeing more discussion on a system analysis based on use requirements and its implication to a system design.*

Thanks for your kind comment. Even though the development solution is based on the existing architecture, open-source libraries and software, this presented tool can be regarded as new in terms of its functionality and ability to interactively calculate loss and risk scenarios through a web-based platform as well as in combination with its supporting decision making module. The background architecture is based on the three-tier client-server architecture model and it was chosen not only for the maintenance but also for the upgrades of the platform at a later time, without needing the users to make changes on the client side. The processing is done mainly on the server and only a web browser is needed for the users to use the platform. For the adopted architecture, we have chosen Boundless (OpenGeo) for several reasons:

- it offers a complete, open-software geospatial software stack with modular components;
- it provides a client-side environment for development of high-level GeoExt-based applications such as developed risk analysis tool or GeoExplorer;
- it allows to implement a faster prototyping of the tool with customizable components, based on the GXP, GeoExt and OpenLayers.
- GXP built-in plugins and widgets make easier to integrate existing tools and functionality in the web-GIS platform. At the same time, it offers the possibility to develop/customize plugins on own specific needs.

In the revised version, we will include a summary, as reviewer 2# also suggested.

2. Partly: description of schema architecture (section 2.3.1) seems incomplete:

- *p. 4016, line 18-19: "supporting data management module" is not illustrated on Fig. 7 (contrary to what authors' claim earlier in the sentence)*

Actually, Fig 7 already includes the schema for the "supporting data management module". However, we missed to mention exclusively (in the caption and text) that the first three tables (hazards, elements-at-risk and vulnerability) belong to the data management module, where all records of the uploaded maps and curve information are stored. The latter three tables (loss, annualized risk and relationship table) belong to the risk analysis module. For better clarification, we will update the explanation and caption of the figure accordingly in the revised version.

- *p. 4017, line 1: not all child 'tables' are present in Fig. 7 – e.g. the "Fella buildings" table is missing. To fix this, I suggest, instead of only a part of it, include the full data model (in the schema design description and in fig. 7) (p. 4016, line 17)*

Regarding the child 'table', e.g. Fella buildings, the user directly uploads the shape files (in zip format) through the interface of the platform (as similar to the hazard interface, Fig. 12). Therefore, this child table is created dynamically within the database upon the upload. This is the reason why it was not present in the fixed schema design and the column attributes of such child tables can be varied depending on the user's uploaded data. Fig 7 thus presents the complete data model of risk analysis and data management module, which is a part of the big data model of the whole platform. We will clarify and explain this in the text.

Technical corrections:

- *p. 4010, line 12: ". . .open source data. ."*
- *p. 4011, line 8: ". . .contribute to the practice of the open-source and research community..."*
- *p. 4011, lines 19-20: ". . .presented on Fig. 1, where the risk analysis tool is one of the main modules." – there is no tool on Fig. 1.*
- *p. 4014, line 3: 'overlain' should be 'overlaid'*
- *p. 4015, lines 9-11: "The batch processing. . .is possible to include in future versions of the platform" – such sentence would be better placed in Section 4 Discussion and conclusion."*
- *p. 4015, line 23: "In this prototype version. . ." would sound better as "In the current version of the prototype. . ."*
- *p. 4017, line 1: the correct spelling of the GeoServer is GeoServer – please check the whole paper for correct spelling of the tool.*
- *p. 4023, line 17 and line 18: ". . .from EUR 3.7 million to 15 million." and ". . .from EUR 0.026 million to 0.4 million. . ."*
- *p. 4028, line 8: "Alexander D.E.:. . .of an unquiet Earth. . . ". Not having done this myself, I suggest authors revise of the whole reference section.*

We thank the reviewer again for providing these corrections and we will correct them in the final paper.

- *p. 4017, lines 21-25: check the proper terminology used for describing geospatial web tools and interfaces. For instance in the sentence: "With the use of GeoServer's import configuration. . ." is unclear. What is the 'GeoServer's import configuration'? With GeoServer, we do not 'import', but 'publish' layers through standard interfaces to the web. We certainly do not 'import layers directly to the database' with the GeoServer – the layers are already in the database (in spatial tables) and can be published with the GeoServer to the web. In case of an interactive tools, GeoServer may allow (through a dedicated interface, such as WFS-T) updating a database. Next in this paragraph, authors emphasize that for importing raster data to a PostgreSQL/PostGIS database an additional application (raster2pgsql) is necessary and suggest that for vector data an import is automatic. However, although it is now obscured from current users of the PostgreSQL/PostGIS database, vector data in a shapefile format is imported to a PostGIS database also through an application (shp2pgsql) additional to PostgreSQL/PostGIS.*

For the proper terminology, we will revise accordingly to avoid the confusion. Using the term 'import' probably confused the explanation of the described process. In the prototype, GeoServer's REST configuration is used to programmatically configure operations such as creating a new feature type or data store in Geoserver. For the uploading of vector and raster data from the client interface, we use an existing GXP (javascript) plugin which allows the user to upload and publish vector (shape files) and raster (tiff) to the GeoServer. The vector layer is stored in the 'postgis' data store and raster is in a coverage store. However, for the loss calculation, we needed the uploaded raster also in the postgis database as the spatial processing is done within the database for this prototype version. Therefore, we specifically coded a php script to further store raster data programmatically in the database using *raster2psql* tool. This was what we meant by '… need to be further imported into the database'.

- *p. 4017-4019: In the description of the processing steps it is unclear when authors refer to in-built functions of the database management system (e.g. ST_Intersects) and when to other parts of the prototype developed especially for the presented interactive web-GIS (e.g. steps for filling attributes of the loss table). This should be clarified.*

Thank for the kind suggestion. We will clarify and include a detailed and better explanation of the processing steps. Firstly, we created a loss table populated with the records of a sub-query, which returned the building ids, its

geometry and a set of pixel values (i.e. min, max and average intensity values) of the polygonised, clipped raster (where the geometry of intensity raster and building layers are intersected). Secondly, the new columns are added to this loss table with respective attribute values (i.e. spatial probability, vulnerability, building type and its monetary value) for those affected buildings, based on given input information. For example, monetary values are extracted from the building layer and spatial probability values can either come from the associated spatial probability map or a given single value entered by the user. After filling all these attribute values in the table, we only need to multiply them as explained in Equation 1 to obtain the total loss of each affected building, and update the table attributes accordingly. As a third step, we registered the record of this newly completed loss table in the 'loss scenario' table as shown in the schema Fig. 7, so that we can retrieve the information later in the system. Finally, we published the calculated loss table (layer) to the GeoServer for visualization in the platform. All these processing steps were carried out automatically through a php server side script, when the user click 'Run' button of Fig. 15 after filling all input parameters in the loss interface. The resulted loss layer is then illustrated in Fig. 17 and it is added automatically into the map upon the successful completion of the loss calculation.

- *p. 4018, lines 19-24: this paragraph sounds as a far too complicated description of exposing spatial layers through web map service interface. Is the "information on existing published layers or styles" the content of WMS's GetCapabiltiies operation? If yes, then, please state it – in section 2.3.2 you indicate that your prototype uses standard interfaces, such as WFS and WMS. These interfaces have standard operations (e.g. GetCapabilities) and results of these operations are in a standard format (e.g. XML). Unlike authors suggest, it is not the GeoServer's REST configuration that ensures the standard operations and their format, it is the standard interface defined by OGC that does it.*

In the final version of the paper, we will revise this paragraph (and above in p. 4017) for clarification. We do agree with the reviewer that standard services such as WMS, WFS and WFS-T, etc. are defined by OGC. The prototype indeed uses such services for operations such as retrieval of the legend graphic, feature information, editing and deleting of features through the web-GIS interface. In order to programmatically configure GeoServer (without using its standard web interface), php-cURL (client URL) and REST configuration are also used. The REST facilitated the process between the client and GeoServer (e.g. in XML format) through HTTP calls like GET, POST or PUT to obtain the information, create or make changes to something in the GeoServer. For example, to add a new style or change the name of a certain published layer (if a user is authorized to do that). This was the case when we publish a new calculated "loss scenario" layer to the GeoServer (i.e. to create a new feature type).

- *p. 4023, line 25: "from 0 to 100. . ." of what? Buildings? ". . .with and increase in return periods of the events." Which return periods – low or high?*

Yes, 0-100 was referred to the number of affected buildings and this number was increased in the higher return periods of the events.

- *p. 4024, line 3: the sentence "if a building was touched to multiple pixels. . .' is confusing. Please rephrase – check the proper terminology (e.g. 'a building was touched to multiple pixels'??) or verify the content.*

Thanks for the suggestion. We will revise the sentence accordingly.

- *p. 4026, lines 23-25: "Moreover, this prototype is as an initial step towards a more complex system as it plays an important role in obtaining feedback and suggestions. . ." – An initial step plays an important role??? Please review and revise the sentence.*

We wanted to highlight that having this prototype is useful and it is an important step as we can present it to the potential users and get the feedback on the application development (from time to time), despite being a prototype version. We will revise it in the final version.