VR-TestCoverage:
Test Coverage Visualization and Immersion in Virtual Reality

Roy Oberhauser
Aalen
Germany

Based on the conference paper in VALID 2022:
"VR-TestCoverage: Test Coverage Visualization and Immersion in Virtual Reality"

Presenter:

- Worked for 14 years in the software industry in the Silicon Valley and in Germany doing research and development.
- Since 2004 he has been a Professor of Computer Science at Aalen University in Germany, teaching in the areas of software engineering.
- His research interest is to leverage technologies and techniques to innovate, automate, support, and improve the production and quality of software for society.

Contents

- Challenge / Problem
- Solution
- Implementation
- Evaluation
- Conclusion

Introduction

- With the increasing pressure to deliver additional software functionality, software engineers and developers are often confronted with the dilemma of "*how much software testing is necessary?*".
- One efficiency aspect to avoid is *test redundancy*
  - I.e., creating further tests in areas already sufficiently tested
- Measuring test (or code or statement) coverage can help focus test development on those areas that are insufficiently tested.
- Software is inherently invisible and its structures (product and test) lack an expressive visual form

As software projects grow, it can become difficult to *visualize*
both the software *product* and the software *tests*
and especially their *dependencies*
in order to "see" what has been addressed and what remains

Challenge

- Considering the adoption rate of test coverage by software developers, for an insight into the industrial popularity of test coverage,
  - Of 512 developers randomly surveyed at Google in a 2019 survey [6], 45% indicate they use it (very) often when authoring a changelist and 25% sometimes.
  - When reviewing a changelist, 40% use coverage (very) often and 28% sometimes.
  - Only 10% of respondents never use coverage.
  - Overall, a substantial number of developers apply code coverage regularly and find value in it.
  - Voluntary adoption at the project level went from 20% in 2015 to over 90% by 2019.
- Yet these relatively high reported rates in professional private companies may not correspondingly be found in smaller less-professional companies or in voluntary development work, e.g., on open source projects.
  - For instance, a survey of 102 open source Android app developers [7] reported that 64% did not use or did not consider code coverage useful for measuring test case quality. Testing has never typically been the forte of software developers.

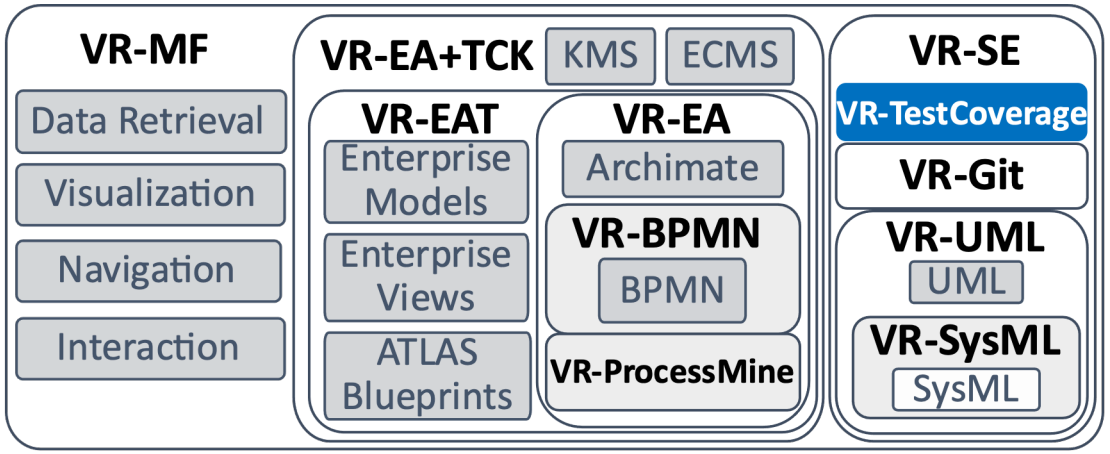
One challenge is how to motivate developers to do testing
and to verify the degree to which their tests cover the product code.

Solution Approach

- Virtual Reality (VR) can play a part in addressing this challenge as a motivational aspect for software testing and for utilizing and expressing test coverage data.
- In our view, an immersive VR experience can be beneficial for a software analysis tasks.
  - Müller et al. [8] compared VR vs. 2D for a software analysis task, finding that VR does not significantly decrease comprehension and analysis time nor significantly improve correctness (although fewer errors were made). While interaction time was less efficient,
  - VR improved the user experience, was more motivating, less demanding, more inventive/innovative, and more clearly structured.
- VR provides an unlimited immersive space for visualizing and analyzing a growing and complex set of system models and their interrelationships simultaneously in a 3D spatial structure viewable from different perspectives.
- As software projects grow in size and complexity, an immersive digital environment:
  - Can provide an additional visualization capability to comprehend and analyze both the software production code (i.e., test target) and the software test suite and how they relate,
  - Can help determine areas where the code coverage achieved by the test suite is below expectations

VR-TestCoverage Solution Concept vs. our other Solution Concepts

- VR-TestCoverage is based on our generalized VR Modeling Framework (VR-MF) (detailed in [11]).
- VR-MF provides a VR-based domain-independent hypermodeling framework addressing four aspects requiring special attention when modeling in VR:
  - visualization, navigation, interaction, and data retrieval.
- Our VR-SE area includes VR-TestCoverage, VR-Git, and VR-UML [1] and VR-SysML [10].
- Since Enterprise Architecture (EA) can encompass SE models and development and be applicable for collaboration in VR, our other VR modeling solutions in the EA area include the aforementioned VR-EA [11], VR-ProcessMine [12], and VR-BPMN [13].
- VR-EAT [14] integrates the EA tool Atlas to provide dynamically-generated EA diagrams in VR
- VR-EA+TCK [14] integrates Knowledge Management Systems (KMS) and/or Enterprise Content Management Systems (ECMS)

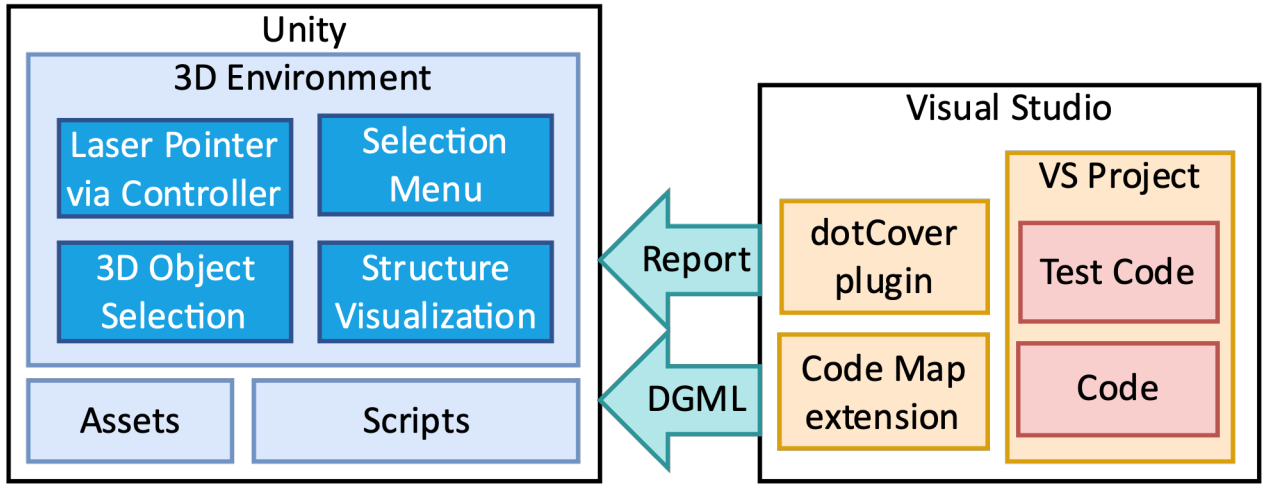


VR-TestCoverage Solution Concept

- Visualization in VR:
  - A *plane* is used to group the production code (test suite target) as well as the test suite.
  - A *tree map using a step pyramid paradigm* (or mountain range) is used to stack containers (i.e., groups, collections, folders, directories, packages) in the third dimension (height) on the plane.
  - One visualization challenge we faced was that we initially thought we could depict the test target code by simply overlaying a layer of it on the production code and indicating which test “covered” which production code.
    - However, once we completed the dependency analysis of large projects we found an n-m relation between tests and the test targets.
  - Thus, we chose to keep the visual depiction of the test suite separated from the test target (since it can have its own hierarchical organization), yet use the same visualization paradigm to depict “containers” or collections as packages or folders.
  - However, to retain the intuitive paradigm of “coverage,” we elected to place the test suite visualization directly above the test target.
    - That way dependencies can be followed from top to bottom (the test target should not depend on any test code).
  - Elevation is used to express concrete tests as “closer” to the target than their containers or abstractions.
  - Dependencies are then shown as lines between the test and test target, analogous to puppet strings
- Navigation in VR
  - The left controller was used for controlling flight (moving the VR camera), while the right controller was used for interaction
- Interaction in VR
  - Since interaction in VR is not yet standardized, in our concept user-element interaction is supported primarily through VR controllers and our VR-Tablet. The VR-Tablet is used to provide context-specific detailed element information. It includes a virtual keyboard for text entry via laser pointer key selection

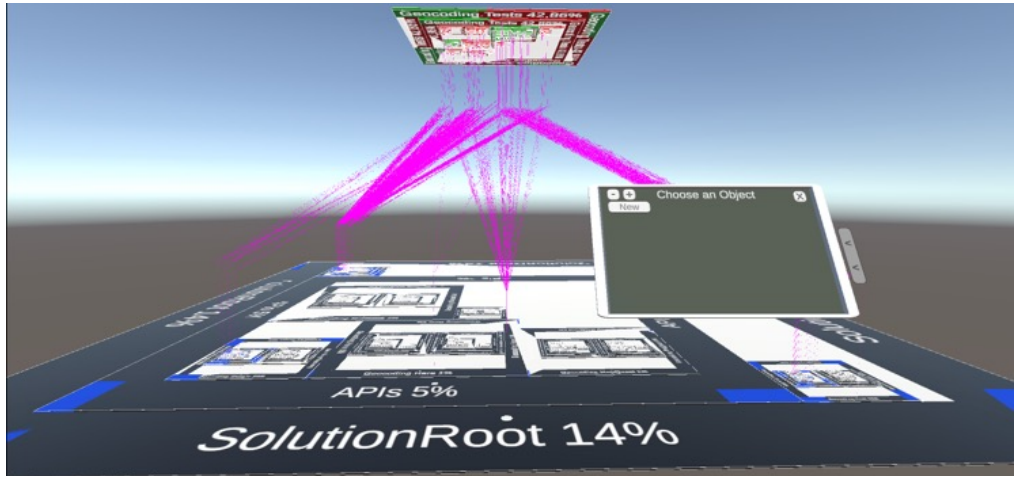
VR-TestCoverage Realization

- Basic visualization, navigation, and interaction functionality in our VR prototype is implemented with Unity 2021.1 and the OpenVR XR Plugin 1.1.4
- JSONUtility library is used for JSON processing
- JetBrains dotCover was used to report test coverage
  - This Microsoft Visual Studio plugin is a .NET Unit test runner und code coverage tool that can generate a statement coverage report in JSON, XML, etc.
- A challenge we faced is that among the coverage tools we considered, they only report on dependencies *between test targets*, and do not explicitly indicate or name direct dependencies from product to the invoking test
- Code Map in Visual Studio 2022 Enterprise Edition can generate a Directed Graph Markup Language (DGML) file for dependencies.
  - After converting it to JSON, we partition the dependency report into a node report and a link report.
  - Only direct dependencies between a test and the product (test target) are then considered.



VR-TestCoverage Realization

- Tests in the test suite (and their containers) are colored based on the **test result** status:
  - Green for successful
  - Red for failed
  - Yellow for other (such as ignored)
- **Coverage** of the test targets is shown as a bar on all four sides and on the elevation area
  - Blue visually indicates the percentage of coverage
  - Black is used for the rest
  - The coverage percentage is also provided numerically
- Dependencies are drawn as magenta lines
- A top-level container is used to represent the overall project.
- Project (test and test target) structures are expressed as containers utilizing a tree map using a 3D step pyramid paradigm
- Our VR-Tablet concept provides detail and interaction

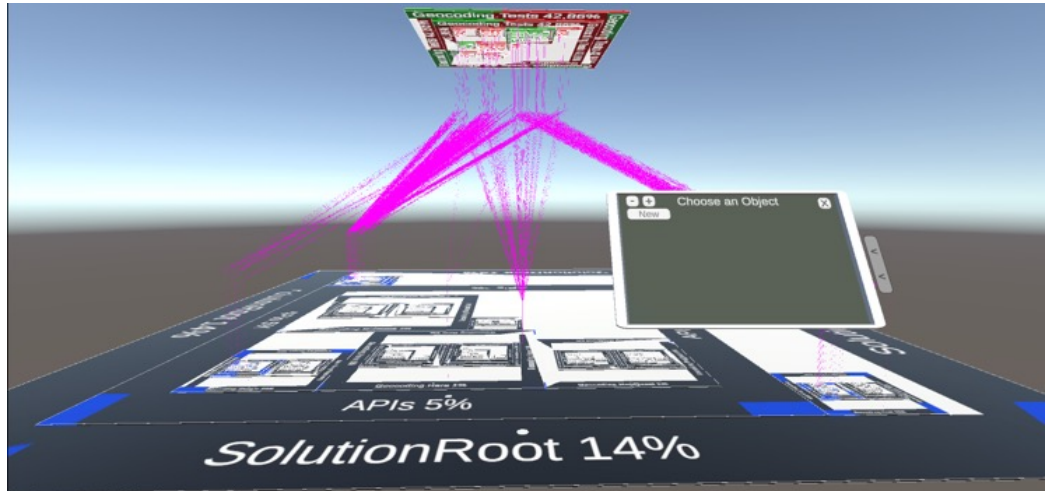


Evaluation

- To evaluate our prototype realization of our solution concept, we use a case study based on three scenarios:
  - Test coverage
  - Test results
  - Test dependencies
- Geocoding.net was used as an example C# project to visualize

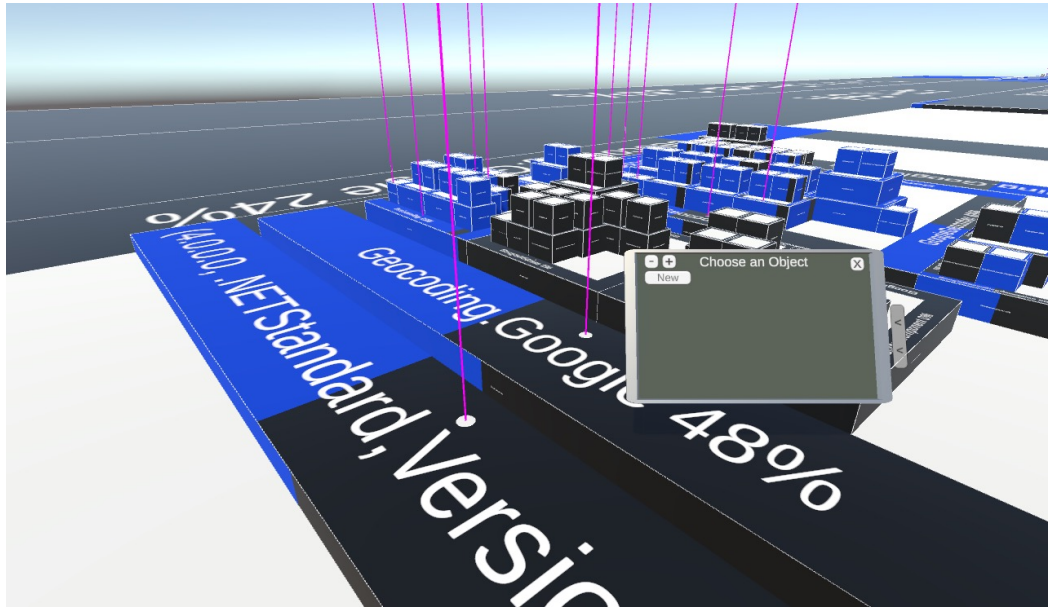
Evaluation: Test Coverage Scenario

- Testers wish to quickly assess
  - the number of test failures (pass/fail rate based on areas) and
  - test coverage, typically concerned about the degree of overall coverage
    - E.g., to compare its level against some high-level test goal,
    - While also concerned about assessing details and risks as to which areas are insufficiently covered by tests.
- Visualization of the system under test (SUT) or test target is shown on a plane, with the coverage percentages for a container (folder, directory, package) shown on each side.
- A top-level container is used to represent the overall project.
- The test suite structure is projected above this onto a separate plane and upside-down
- Test target and code coverage are on the bottom, test suite and test results are visible on top
- The VR-Tablet is shown on the right
- Dependencies are seen via magenta lines



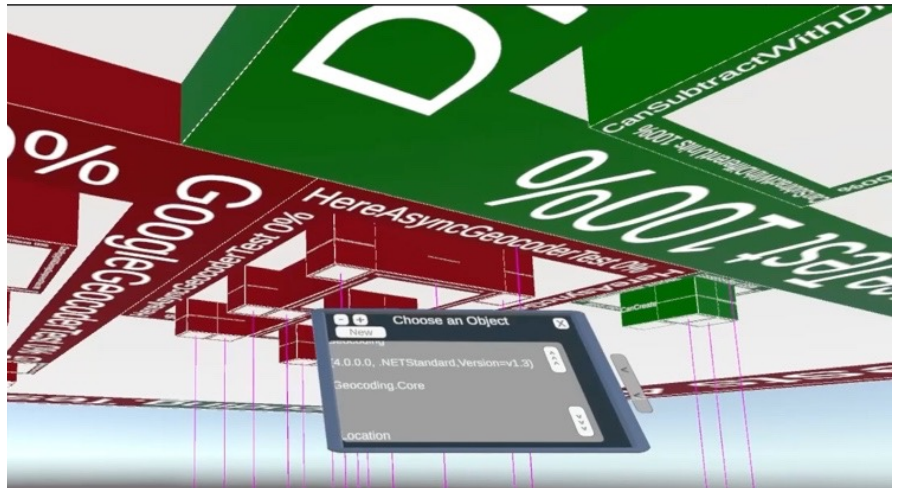
Evaluation: Test Coverage Scenario

- Test coverage of the test targets is indicated via a bar on all four sides so that from any perspective the coverage is visually indicated.
- A bar graph is used on all sides, with blue visually indicating the percentage of coverage and black used for the rest (the exact coverage percentage is also shown numerically).
- A stepped pyramid paradigm is used to portray the granularity, with the highest cubes having the finest granularity or depth, and the lowest being the least granular.
- For instance, a user can quickly hone in on overall areas with little to no blue, meaning that coverage there was scarce, and one can quickly find and focus on details (without losing the overview) by focusing on the higher elevations.



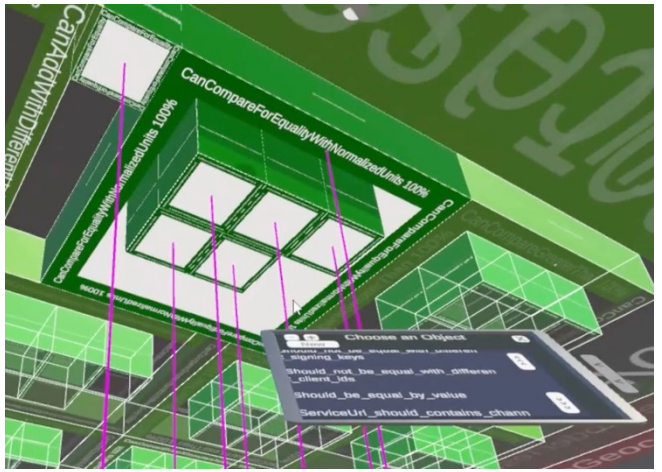
Evaluation: Test Results Scenario

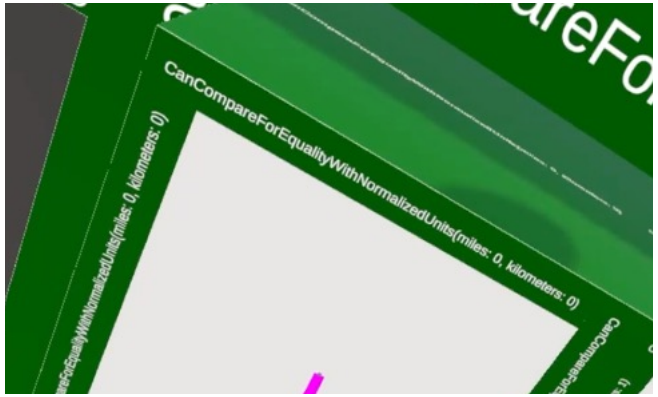
- If tests have been run, besides coverage, a tester is also typically interested in the test results and overall pass (or success) rate.
- We visualize the test suite as a tree map of all tests using a step pyramid for the third dimension to indicate granularity via depth. Analogously to how coverage was shown as a colored bar on all four sides of a container, on the test suite green is proportionally shown for success rate and red for failure (yellow for other), with its numerical value also given.
- On the right is a closeup, showing that the test case and unit test information is provided.
- Our virtual VR-Tablet concept permits one to inspect detailed test results for a selected test



Evaluation: Test Dependency Scenario

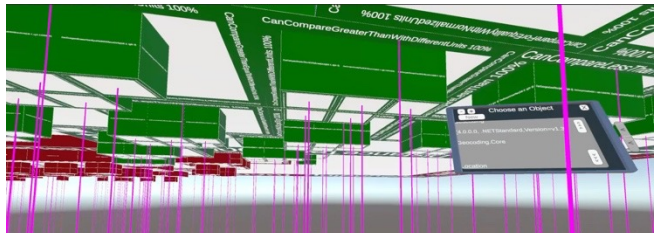
- One scenario that is often unavailable for testers is a dependency view, with which they can view which tests are directly invoking or reaching which target code.
- Typically, by convention tests are named in such a way to express the test target, yet the dependencies could nevertheless differ from what one might expect.
  - This is especially true if the test suite consists not only of unit tests but also integration or system tests.
- By eliminating the guess work, dependencies could be used to determine which tests are primarily reaching a target, and then focus on extending that test in order to increase the target coverage.
- One challenge is that there is not necessarily a 1-1 match of test to test target, thus dependency links provide a way to visualize these hitherto hidden dependencies.
- Selecting a test target, the test dependencies can be followed via a magenta line; the associated test cases can be seen in the test suite (upper figure) and can be followed to the most granular level of the test case (lower figure).
  - Unassociated tests are then ghosted.





Evaluation: Test Dependency Scenario

- The VR-Tablet (top right) can be used to inspect test report details about a selected test object
  - Here showing the test method (CanCompareForEqualityWithNormalizedUnits), test data input values (miles: 1, kilometers: 1.609344), test status (success).
- The magenta links can be followed to the test target plane to determine what target a selected test is directly reaching (middle figure).
- By unselecting a test element, all dependencies are shown again with all test elements opaque (bottom figure).
  - Thus, one can follow overall groups or determine that certain tests are perhaps prepared and not (as yet) linked or related to the test target, since no dependencies to the target are shown.
  - This could occur if tests are written before the production code has been implemented, which can be expected, for instance when applying test-driven techniques.
  - Alternatively, this could be an indicator of a test suite and test target mismatch, perhaps if the production code was significantly changed without making associated changes to the test suite.



VR-TestCoverage Conclusion

- VR-TestCoverage contributes an immersive test coverage experience for visually depicting and navigating both tests and test targets in VR.
- Our solution concept visualizes the system under test (production code or test target) on a plane using a tree map with a step pyramid paradigm.
- The test suite is analogously depicted above it and the dependency links shown.
- The implemented VR prototype demonstrated its feasibility.
- The solution concept was then evaluated using our protype based on a case study involving three scenarios:
  - Test coverage, test results, and test dependencies.
- The evaluation results showed that all three scenarios are supported by our solution concept and its realization.

VR-TestCoverage provides an immersive experience for test coverage metrics and reports, and can enhance and provide a motivational aspect to the testing process in general.

References

1. C. Metz, “Google Is 2 Billion Lines of Code—And It’s All in One Place,” 2015. [Online]. http://www.wired.com/2015/09/ google-2-billion-lines-codeand-one-place/
2. Evans Data Corporation. [Online]. https://evansdata.com/press/viewRelease.php?pressID=293 2022.06.10
3. “Software Engineering — Guide to the software engineering body of knowledge (SWEBOK),” ISO/IEC TR 19759:2015, 2015.
4. "ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing--Part 4: Test techniques," ISO/IEC/IEEE 29119-4:2015, 2015, doi: 10.1109/IEEESTD.2015.7346375.
5. L. Inozemtseva and R. Holmes, “Coverage is not strongly correlated with test suite effectiveness,” In Proceedings of the 36th international conference on software engineering (ICSE 2014), ACM, 2014, pp. 435-445.
6. M. Ivanković, G. Petrović, R. Just, and G. Fraser, “Code coverage at Google,” In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, 2019, pp. 955-963.
7. M. Linares-Vásquez, C. Bernal-Cardenas, K. Moran and D. Poshyvanyk, "How do Developers Test Android Applications?," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2017, pp. 613-622, doi: 10.1109/ICSME.2017.47.
8. R. Müller, P. Kovacs, J. Schilbach, and D. Zeckzer, "How to master challenges in experimental evaluation of 2D versus 3D software visualizations," In: 2014 IEEE VIS International Workshop on 3Dvis (3Dvis), IEEE, 2014, pp. 33-36
9. R. Oberhauser, “VR-UML: The unified modeling language in virtual reality – an immersive modeling experience,” International Symposium on Business Modeling and Software Design, Springer, Cham, 2021, pp. 40-58.
10. R. Oberhauser, “VR-SysML: SysML Model Visualization and Immersion in Virtual Reality,” International Conference of Modern Systems Engineering Solutions (MODERN SYSTEMS 2022), IARIA, 2022. To be published.
11. R. Oberhauser and C. Pogolski, "VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN," In: Shishkov, B. (ed.) BMSD 2019. LNBIP, vol. 356, Springer, Cham, 2019, pp. 170–187.
12. R. Oberhauser, “VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality,” The Fourteenth International Conference on Information, Process, and Knowledge Management (eKNOW 2022), IARIA, 2022. To be published.
13. R. Oberhauser, C. Pogolski, and A. Matic, "VR-BPMN: Visualizing BPMN models in Virtual Reality," In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, Springer, Cham, 2018, pp. 83–97. https://doi.org/10.1007/978-3-319-94214-8_6
14. R. Oberhauser, P. Sousa, and F. Michel, "VR-EAT: Visualization of Enterprise Architecture Tool Diagrams in Virtual Reality," In: Shishkov B. (eds) Business Modeling and Software Design. BMSD 2020. LNBIP, vol 391, Springer, Cham, 2020, pp. 221-239. https://doi.org/10.1007/978-3-030-52306-0_14
15. R. Oberhauser, M. Baehre, and P. Sousa, “VR-EA+TCK:
Visualizing Enterprise Architecture, Content, and Knowledge
in Virtual Reality,” In: Shishkov, B. (ed.) BMSD 2022, LNBIP, Springer, Cham, 2022. To be published.
16. J. Vincur, P. Navrat, and I. Polasek, “VR City: Software analysis in virtual reality environment,” In 2017 IEEE international conference on software quality, reliability and security companion (QRS-C), IEEE, 2017, pp. 509-516.
17. K. Dreef, V. K. Palepu and J. A. Jones, "Global Overviews of Granular Test Coverage with Matrix Visualizations," 2021 Working Conference on Software Visualization (VISSOFT), 2021, pp. 44-54, doi: 10.1109/VISSOFT52517.2021.00014.
18. A. Rahmani, J.L. Min, and A. Maspupah, “An evaluation of code coverage adequacy in automatic testing using control flow graph visualization,” In 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE), IEEE, 2020, pp. 239-244.
19. F. Pecorelli, G. Di Lillo, F. Palomba, and A. De Lucia, “VITRuM: A plug-in for the visualization of test-related metrics,” In Proceedings of the International Conference on Advanced Visual Interfaces (AVI ’20), ACM, 2020, pp. 1-3, https://doi.org/10.1145/3399715.3399954
20. K. Alemerien and K. Magel, “Examining the effectiveness of testing coverage tools: An empirical study,” International journal of Software Engineering and its Applications, 8(5), 2014, pp.139-162.
21. K. Sakamoto, K. Shimojo, R. Takasawa, H. Washizaki and Y. Fukazawa, "OCCF: A Framework for Developing Test Coverage Measurement Tools Supporting Multiple Programming Languages," 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, 2013, pp. 422-430, doi: 10.1109/ICST.2013.59.