ID	Article Title	Author(s)	Advantages (RQ1)	Disadvantages (RQ2)	Static tools (RQ3)	Solutions (RQ4)					
1	STATIC CODE ANALYSIS TOOLS: A SYSTEMATIC LITERATURE REVIEW.	Stefanović, Darko Nikolić, Danilo Dakić, Dušanka Spasojević, Ivana Ristić, Sonja	Manual code review is time-consuming. Optimizing the operation of the compiler. Detecting irregularities. Help understand the behaviour of a program without execution.	Defects must be visible in the source code. Not all programming languages are supported. A single tool might not be able to detect all defects.	Cppcheck, FindBugs, Splint, SonarQube, PMD, Flawfinder	Range of programming languages supported is continuously expanding. Combining tools can help detect all defects.					
2	Probing into code analysis tools: A comparison of C# supporting static code analyzers	Shaukat, Rida Shahoor, Arooba Urooj, Aniqa	Saves enourmous amount of time and money in a development process. Some tools are highly customizable.	Tool can give vague explanations and unwanted checks. High memory consumption (ReSharper, Parasoft).	FxCop, NDepend, Nitriq, ReSharper, Coverity Scan, PVS-studio, Parasoft dotTest, Visual Code Grepper	Correct tools have technical and accurate descriptions. Some tools can be customized to fix vague problems.					
3	A Comparison of Open-Source Static Analysis Tools for Vulnerability Detection in C/C++ Code	Arusoaie, Andrei Ciobaca, Stefan Craciun, Vlad Gavrilut, Dragos Lucanu, Dorel		Some tools do not detect all defects. Difficult on install/compile some tools because they rely on older versions of libraries of compilers. For some defect types there is no good tool.	Clang, Frama-C, Oclint, Cppcheck, Splint, Infer, Uno, Flawfinder, Sparse, Flint++	Certain tools are better at detecting certain defect types than others.					
4	How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines	Zampetti, F. Scalabrino, S. Oliveto, R. Canfora, G. Di Penta, M.	Early detection of potential faults, vulnerabilitues, code smells.		CheckStyle, FindBugs, PMD, Apache-rat, Clirr, jDepend						
5	Static Code Analysis Tools with the Taint Analysis Method for Detecting Web Application Vulnerability	Maskur, Achmad Fahrurrozi Dwi Wardhana Asnar, Yudistira	Tools help with the quality of the code as an engineer might not know the best methods (especially security-wise) and therefore also give insight to the user. Vulnerabilities can be found without executing.	Some vulnerabilities are not found. False negatives.		Detecting vulnerabilites using taint analysis can be improved by supporting OOP. Tools are to be improved based on false negatives and manually discovered defects that were unfound by a tool.					
6	Using Machine Learning Techniques to Classify and Predict Static Code Analysis Tool Warnings	Alikhashashneh, Enas A. Raje, Rajeev R. Hill, James H.	Examine code without execution.	False positives. False negatives.		Machine learning (particularily Random Forests technique) can help reduce false positives. Also techniques like KNN, SVM and RIPPER. False pos/neg can also be avoided if developers rewrite their code in a way that reduces source code complexity, coupling and usage of global variables.					
7	Comparative study on static code analysis tools for C/C++	Fatima, Anum Bibi, Shazia Hanif, Rida	Efficient system to check on software coding scheme. Defects (safety, security bugs, quality standards, dereferences, buffer overruns, injection problems, memory leaks, dataflow problems, language implementation errors, inconsistencies) are detected without execution. Helps build long-lasting software without bugs and vulnerabilites.	No tool can give 100% surety that software will never halt, crash, misperform. False negatives. Some tools are too specific and lack basic checks, others are just too basic.	FlawFinder, VCG, CppCheck, Splint, Polyspace, CSTAT (was the best of this bunch), QAC, Coverty, Astree, Klocwork, Parasoft, YASCA, Sparse, ITS4, Goanna, RATS	Tools can be improved as lacking functionality is discovered. Tools can be used together to cover all important code vulnerabilities.					
8	Evaluating how static analysis tools can reduce code review effort	Singh, Devarshi Sekar, Varun Ramachandra Stolee, Kathryn T. Johnson, Brittany	Tools can help to reduce code review effort. Code reviewer or developer does not need to know the best practice patterns. Tools automatically find defects and style issues. No need for reviewers to use comments -> less noise.	False positives.	PMD, FindBugs, CheckStyle	Often the best tools (etc PMD) do not have high amount of false positives. For the remaining false positives, configuring a tool's (PMD's) ruleset can limit the warnings.					
9	Identifying and Documenting False Positive Patterns Generated by Static Code Analysis Tools	Reynolds, Z.P. Jayanth, A.B. Koc, U. Porter, A.A. Raje, R.R. Hill, J.H.	Tools help can help developers find defects automatically. Manual code analysis is time-consuming.	False positives. False negatives.	CAT.NET, FindBugs	Static code analysis tools can be integrated into frameworks for evaluating the tools which can help identify potential false positives. Reduce usage of some techniques like global variables to avoid some false positives. False positives can still be "fixed". False positive patterns can be studied and fixed. Multiple tools can be used together.					
10	A Novel Memory Leak Classification for Evaluating the Applicability of Static	Zhang, Sen Zhu, Jingwen Liu, Ao Wang, Weijing Guo, Chenkai Xu, Jing	Tools can detect memory leaks.	Difficult to design a tool that can detect all kinds of memory leaks. Complex problems are unfound by even the best tools. Hard to find the absolute best tool as all tools have advantages over others.	Cppcheck, Clang Static Analyzer, Sourceinsightscan	Choosing the correct tool for your situation can improve performance.					

11	Prioritizing Alerts from Multiple Static Analysis Tools, Using Classification Models	Flynn, Lori Snavely, William Svoboda, David VanHoudnos, Nathan Qin, Richard Burns, Jennifer Zubrow, David Stoddard, Robert Marce-Santurio, Guillermo	Tools can help determine flaws without executing the code.	False positives. False negatives. Complex control flow or data flow constructs significantly reduce tools' success rate.		Machine learning can help with false positives. Tools can be used together for improved efficiency.					
12	Software quality through the eyes of the end-user and static analysis tools	Kamonphop Srisopha, Reem Alfayez	The analysis tools can reveal possible vulnerabilities, defects, or design issues at an early stage in the development phase		PMD, FindBugs, SonarQube						
13	Scientific Developers v/s Static Analysis Tools: Vision and Position Paper	Rohan Krishnamurthy, Thomas S. Heinze, Carina Haupt, Andreas Schreiber, and Michael Meinel	They help identify defects and code smells or enforce common coding standards.	High false positives rates, low comprehensibility of analysis results, and missing process integration restrain the use and acceptance of static analysis tools.							
14	A Practical Approach for Ranking Software Warnings from Multiple Static Code Analysis Reports	Binh Hy Dang		The problem of using multiple bugs finding tools is they not only detect similar software defects but also generate new warning messages. The excessive warnings make code analysis time consuming and expensive. High number of false positive.	FindBugs, PMD, Cppcheck, Understand	Static analysis warnings can be prioritized by using a data fusion technique that merges warnings from different tools and reports them in a universal format or using PCA to develop an analytical model and rank the alerts.					
15	Automatically Generating Fix Suggestions in Response to Static Code Analysis Warnings	Diego Marcilio, Carlo A. Furia, Rodrigo Bonifácio, Gustavo Pinto	Using these tools can help developers monitor and improve software code quality	They report warnings that may always not correspond to an actual mistake. Developers normally fix only a small fraction (typically, less than 10%) of the reported issues.	SonarQube, FindBugs, SpotBugs	If these analysis tools could automatically provide suggestions on how to fix the issues that trigger some of the warnings, their feedback would become more actionable and more directly useful to developers					
16	Source Code Analysis for Secure Programming Practices	Christian Barrientes , Jeong Yang , Joshua Sanchez , and Young Rae Kim	Static analysis of source code can help mitigate the common coding errors such as buffer overflow, memory leaks, unused variables, and various race conditions		FindBugs, Find Security Bugs						
17	Challenges with Responding to Static Analysis Tool Alerts	Nasif Imtiaz, Akond Rahman, Effat Farhana, and Laurie Williams	Help developers detect potential defects in the code early in the development cycle	False positive alerts, the way in which the alerts are presented, incomprehensible and untrustworthy alerts and a lack of customizability	FindBugs	Alert messages should be more effective in helping the developers act on the alert					
18	An Empirical Assessment of Machine Learning Approaches for Triaging Reports of a Java Static Analysis Tool	Ugur Koc, Shiyi Wei, Jeffrey S. Foster, Marine Carpuat, Adam A. Porter		High false positive rates	FindSecBugs	Machine learning techniques have been proposed for false positive detection: hand-engineered features, bag of words, recurrent neural networks, and graph neural networks, for classifying false positives					
19	How Developers Diagnose Potential Security Vulnerabilities with a Static Analysis Tool	Justin Smith , Brittany Johnson, Emerson Murphy-Hill , Bill Chu, and Heather Richter Lipford	These tools locate and report on potential software security vulnerabilities, such as SQL injection and cross-site scripting even before the code executes.	Researchers cite several related reasons why these tools do not help developers resolve defects, for instance, the tools: “may not give enough information”; produce “bad warning messages”; and “miscommunicate” with developers.	Find Security Bugs	Most broadly, tools should support effective strategies and provide information that aligns with the information needs we have identified. Tools should help developers, among other things, search for relevant web resources.					
20	The Use and Limitations of Static-Analysis Tools to Improve Software Quality	Paul Anderson	The latest static analysis tools are capable of finding serious errors in programs such as null-pointer de-references, buffer overruns, race conditions, resource leaks, and other errors. Static analysis can be used very early in the development cycle, its use can reduce the cost of development. They also make it easier to achieve full code coverage.	There are path limitations, most tools ignore recursive calls and functioncalls that are made through function pointers. Tool's won't give accurate results when parts of the source code are missing or where the fundamental rules of the language aren't being violated		Tools should be used only to compliment other review and testing techniques.					

21	Static Analysis Techniques and Tools: A Systematic Mapping Study	Vinícius Rafael Lobo de Mendonça, Cássio Leonardo Rodrigues, Fabrízzio Alphonsus A. de M. N. Soares, Auri Marcelo Rizzo Vincenzi	Static analysis tools are very useful for carrying out initial verification and validation activities compared to other quality assurance procedures, especially due to their low implementation cost.	The main disadvantage of static analysis tools is their high false positive rates.False positive alerts lead to an increase in process costs, because their detection is usually done by human intervention. This consumes precious time that could be used for the correction of real faults.	FindBugs, PMD, CheckStyle, SonarQube	Static analysis tools should be combined together or with dynamic tool to reduce false positive rates. Another way would be assigning a weight or priority for each type of false positive and so bugs with higher priority would be given greater attention.					
22	A Critical Comparison on Six Static Analysis Tools: Detection, Agreement, and Precision	Valentina Lenarduzzi, Savanna Lujan, Nyyti Saarimäki, Fabio Palomba	There is a wide range of tools today to choose from and they are becoming increasingly easier to use, especially in continuous integration pipelines	Precision is quite low on average	SonarQube, Better Code Hub, Coverity Scan, Findbugs, PMD, and CheckStyle	Different tools should be combined to achieve better coverage. Most problems related to static code analysis tools could be avoided if current tools would be complemented with effective tools targeting automated refactoring and automatic test case generation.					
23	Taxonomy of Static Code Analysis Tools	Jernej Novak, Andrej Krajnc, Rok Žontar	The tools are fast, the code can be inspected much more frequently and they can contain the same level of knowledge as a human reviewer. They can help us find common software errors such as memory overruns, cross-site scripting attacks, injections and various other boundary cases	A tool will never find any error if this behavior has not been specified with rules or patterns. They cannot verify the design, architectural errors, poorly made cryptographic libraries, inappropriately selected algorithms, design problems which cause confusion. They can't find passwords or magic numbers in the code. They are prone to false positives. They are only as good as the rules they are using to scan with.	StyleCop, Gendarme, CheckStyle, FindBugs	They should be used in correlation with manual code analysis and other review tools.					
							· RQ1: What are the main advantages to the usage of static tool analysers				
							· RQ2: What are the main disadvantages to the usage of static tool analysers				
							· RQ3: Which static analysis tools are more used in studies in the last 10 years				
							· RQ4: How can the problems related to the static tool analysers be minimized				