CS 499/599 Research Methods for Software Engineering

Research Project

D4.1. Systematic Literature Review Protocol and Pilot Study

1. Research questions

1.1 Main study research questions

Our main study research questions are as follows:

RQ 1: How does deprecated packages affect the health of project dependencies?

RQ 2: What are the compatibility issues involved in the upgradation of project dependencies in the future?

1.2 SLR research questions

RQ 3: How prevalent are dependency issues?

2. Search strategy

2.1 Resources to be searched

The primary studies for our literature review started with the GitHub Repositories and few research papers gathered from following sources :

- Google Scholar (https://scholar.google.com/)
- Springer Link Online Library (https://link.springer.com/)
- IEEE Xplore Digital Library (https://ieeexplore.ieee.org/)

2.2 Main concepts and synonyms

Reference	Category	Keywords
C1	Software Management and Packages	Software Maintenance, Dependency Management, Software Ecosystems, Computer Bugs, Practice-Driven, Post Development Issues, Code Reusability, Reusable Modules.
C2	Dependency Analysis	Dependency Analysis, Dependency Smells, Incompatible Packages, Outdated Packages, Deprecated Packages, Managing Dependencies.

2.3 Query string

("Software Maintenance" OR "Software Ecosystems" OR "Computer Bugs" OR "Post Development Issues" OR "Code Reusability" OR "Package Manager" OR "Practice-Driven") AND ("Dependency Analysis" OR "Dependency Smells" OR "Incompatible Packages" OR "Outdated Packages" OR "Deprecated Packages")

2.4 Golden set for the validation of the search string

1. A. J. Jafari, D. E. Costa, R. Abdalkareem, E. Shihab and N. Tsantalis, "Dependency Smells in JavaScript Projects," in IEEE Transactions on Software Engineering, vol. 48, no. 10, pp. 3790-3807, 1 Oct. 2022, doi: 10.1109/TSE.2021.3106247.
2. Y. Cao, L. Chen, W. Ma, Y. Li, Y. Zhou and L. Wang, "Towards Better Dependency Management: A First Look At Dependency Smells in Python Projects," in IEEE Transactions on Software Engineering, 2022, doi: 10.1109/TSE.2022.3191353.
3. Callo Arias, T.B., van der Spek, P. & Avgeriou, P. A practice-driven systematic review of dependency analysis solutions. Empir Software Eng 16, 544–586 (2011). https://doi.org/10.1007/s10664-011-9158-8
4. Decan, A., Mens, T. & Grosjean, P. An empirical comparison of dependency network evolution in seven software packaging ecosystems. Empir Software Eng 24, 381–416 (2019). https://doi.org/10.1007/s10664-017-9589-y
5. F. R. Cogo, G. A. Oliva and A. E. Hassan, "Deprecation of Packages and Releases in Software Ecosystems: A Case Study on NPM," in IEEE Transactions on Software Engineering, vol. 48, no. 7, pp. 2208-2223, 1 July 2022, doi: 10.1109/TSE.2021.3055123.

3.Selection criteria

3.1 Inclusion criteria

Our research topic mainly focuses on the analysis of project dependencies and its internal dependencies. It addresses a problem commonly faced by many developers/managers which deals with the problems related to package dependency upgradation or deprecation. We have considered the research papers in this category which have the similar ideology i.e., papers related to dependency analysis. Then we have refined the results using the Articles and English filters.

The inclusion research papers are mentioned below:

1. A. J. Jafari, D. E. Costa, R. Abdalkareem, E. Shihab and N. Tsantalis, "Dependency Smells in JavaScript Projects," in IEEE Transactions on Software Engineering, vol. 48, no. 10, pp. 3790-3807, 1 Oct. 2022, doi: 10.1109/TSE.2021.3106247.
2. Callo Arias, T.B., van der Spek, P. & Avgeriou, P. A practice-driven systematic review of dependency analysis solutions. Empir Software Eng 16, 544–586 (2011). https://doi.org/10.1007/s10664-011-9158-8

3. B. Chinthanet, R. G. Kula, T. Ishio, A. Ihara, and K. Matsumoto. On the lag of library vulnerability updates: An investigation into the repackage and delivery of security fixes within the npm JavaScript ecosystem. arXiv preprint arXiv:1907.03407, 2019.
4. J. Cox, E. Bouwers, M. Van Eekelen, and J. Visser. Measuring dependency freshness in software systems. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 2, pages 109–118. IEEE, 2015.
5. A. Decan and T. Mens, "What Do Package Dependencies Tell Us About Semantic Versioning?," in IEEE Transactions on Software Engineering, vol. 47, no. 6, pp. 1226-1240, 1 June 2021, doi: 10.1109/TSE.2019.2918315.
6. A. Decan, T. Mens, and M. Claes. An empirical comparison of dependency issues in OSS packaging ecosystems. In 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pages 2–12. IEEE, 2017.
7. C. Artho, K. Suzaki, R. Di Cosmo, R. Treinen, and S. Zacchiroli. Why do software packages conflict? In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, pages 141–150. IEEE Press, 2012.
8. C. Mao, "Visualization and Dependency Analysis for Linkage Structures in Web Applications," 2009 Fourth International Conference on Internet and Web Applications and Services, 2009, pp. 119-124, doi: 10.1109/ICIW.2009.106.

3.2 Exclusion criteria

There are a few research papers we excluded during our research process. The criteria being:

1. Paper discussing on software challenges, architectures, approaches, and software library adoption are excluded.
2. Paper explaining regarding the efficient software ecosystem establishment or comparing software ecosystems are excluded.
3. Papers which focus on the dependency networks are excluded.
4. Papers which stated the impact of interlanguage dependencies are excluded.

The Exclusion criteria research papers are :

1. Eghan, E.E., Alqahtani, S.S., Forbes, C. et al. API trustworthiness: an ontological approach for software library adoption. Software Qual J 27, 969–1014 (2019). https://doi.org/10.1007/s11219-018-9428-4
2. Blanthorn, O.A., Caine, C.M. & Navarro-López, E.M. Evolution of communities of software: using tensor decompositions to compare software ecosystems. Appl Netw Sci 4, 120 (2019). https://doi.org/10.1007/s41109-019-0193-5
3. Decan, A., Mens, T. & Grosjean, P. An empirical comparison of dependency network evolution in seven software packaging ecosystems. Empir Software Eng 24, 381–416 (2019). https://doi.org/10.1007/s10664-017-9589-y
4. M. Grichi, M. Abidi, F. Jaafar, E. E. Eghan and B. Adams, "On the Impact of Interlanguage Dependencies in Multilanguage Systems Empirical Case Study on Java Native Interface Applications (JNI)," in IEEE Transactions on Reliability, vol. 70, no. 1, pp. 428-440, March 2021, doi: 10.1109/TR.2020.3024873.

3.3 Selection Process

The steps involved in order to select the papers:

- First apply the query string
- Segregate it based on inclusion and exclusion criteria.
- Further, we would consider the inclusion papers which are quite like our research topic and research questions**.**

4.Data extraction

4.1 Data to be collected

- From each of the selected papers, we would be extracting the challenges faced due to the dependencies.
- We would also be collecting the information like target language, dependency manager, which is taken as reference, severity of the dependency issues etc.

4.2 Data synthesis

The data obtained from the selected papers would be further analyzing based on following questions:

- In which areas are these dependencies are widely used?
- Which target language have this dependency issues?
- What is the root cause of these issues?
- Which users are facing these issues?

5.Pilot study

5.1 Papers Selected

Reference	Status	Reason
A practice-driven systematic review of dependency analysis solutions	Included	The paper talks about the development of methods and techniques to understand and analyze software systems
Dependency Smells in JavaScript Projects	Included	In this paper, we empirically examine evidence of recurring dependency management issues (dependency smells)
Why Do Developers Use Trivial Packages? An Empirical Case Study on NPM	Excluded	The paper describes how developers are concerned about maintenance and the risks of breakages due to the extra dependencies which deviates from our research topic.

5.2 Data collection

Paper Reference	Target Language	Dependency Manager/Reference tool	Challenges discussed
A. J. Jafari, D. E. Costa, R. Abdalkareem, E. Shihab and N. Tsantalis, "Dependency Smells in JavaScript Projects," in IEEE Transactions on Software Engineering, vol. 48, no. 10, pp. 3790-3807, 1 Oct. 2022, doi: 10.1109/TSE.2021.3106247.	JavaScript	NPM	Latest features and fixes ensuring backwards compatibility
Callo Arias, T.B., van der Spek, P. & Avgeriou, P. A practice-driven systematic review of dependency analysis solutions. Empir Software Eng 16, 544–586 (2011). https://doi.org/10.1007/s10664-011-9158-8	Java and C++	Configuration repositories	Software dependencies in software intensive systems.
B. Chinthanet, R. G. Kula, T. Ishio, A. Ihara, and K. Matsumoto. On the lag of library vulnerability updates: An investigation into the repackage and delivery of security fixes within the npm JavaScript ecosystem. arXiv preprint arXiv:1907.03407, 2019.	JavaScript	NPM	Lags in non-related updates and lags in patch fixing.
J. Cox, E. Bouwers, M. Van Eekelen, and J. Visser. Measuring dependency freshness in software systems. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 2, pages 109–118. IEEE, 2015.	General	System-level metric based on an industry benchmark.	Investigate the relationship between outdated dependencies and security vulnerabilities
A. Decan and T. Mens, "What Do Package Dependencies Tell Us About Semantic Versioning?," in IEEE Transactions on Software Engineering, vol. 47, no. 6, pp. 1226-1240, 1 June 2021, doi: 10.1109/TSE.2019.2918315.	Java	Cargo, npm, Packagist and Rubygems	Dependencies version constraints.

A. Decan, T. Mens, and M. Claes. An empirical comparison of dependency issues in OSS packaging ecosystems. In 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pages 2–12. IEEE, 2017.	Rust, Perl, R, JavaScript, .NET, PHP, Ruby	Cargo for Rust, CPAN for Perl, CRAN for R, npm for JavaScript, NuGet for the .NET platform, Packagist for PHP, and RubyGems for Ruby	Impact of existing package dependencies.
7. C. Artho, K. Suzaki, R. Di Cosmo, R. Treinen, and S. Zacchiroli. Why do software packages conflict? In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, pages 141–150. IEEE Press, 2012.	General	Github repositories(The Debian bugrepository and Red Hat's bugzill)	Case study of conflict defects of the packages.
C. Mao, "Visualization and Dependency Analysis for Linkage Structures in Web Applications," 2009 Fourth International Conference on Internet and Web Applications and Services, 2009, pp. 119-124, doi: 10.1109/ICIW.2009.106.	General	WASViewer	Dependency graph which validates the effectiveness of the methods followed.

5.3 Data synthesis

As per our pilot study, we have analyzed the challenges listed in each paper and observed that Java, JavaScript, .Net are the languages where these dependencies are most widely used. Out of which, JavaScript has most common dependency issues. Most of these dependency issues in JavaScript arises due to the improper package versions constraints, no proper patch fixing and no proper information delivery to developers who are dependent on these packages.