Onderzoeksvoorstel
Artificial Intelligence Dependency Checker.
Bachelorproef, 2025-2026

Jelle Vandriessche

E-mail: jelle.vandriessche@student.hogent.be
Project repo: https://github.com/jellev00/hogent-bachproef-paper-jelle-vandriessche
Co-promotor: Nog aan het zoeken naar een Co-promotor

Samenvatting

Hier schrijf je de samenvatting van je voorstel, als een doorlopende tekst van één paragraaf. Let op: dit is geen inleiding, maar een samenvattende tekst van heel je voorstel met inleiding (voorstelling, kaderen thema), probleemstelling en centrale onderzoeksvraag, onderzoeksdoelstelling (wat zie je als het concrete resultaat van je bachelorproef?), voorgestelde methodologie, verwachte resultaten en meerwaarde van dit onderzoek (wat heeft de doelgroep aan het resultaat?).

Keuzerichting: AI & Software Engineering
Sleutelwoorden: AI-agent, dependency management, software maintenance, technische schuld, geautomatiseerde updates

Inhoudsopgave

1. Inleiding

1.1. Probleemstelling

Softwareprojecten binnen bedrijven en organisaties maken gebruik van talrijke externe bibliotheken, frameworks en tools, de zogenaamde *dependencies*. Deze dependencies worden voortdurend bijgewerkt om nieuwe functionaliteiten, verbeterde prestaties en vooral beveiligingspatches aan te bieden (Pashchenko e.a., 2018).

Het manueel opvolgen van deze updates vormt echter een groot probleem in softwareontwikkeling. Ontwikkelaars moeten regelmatig changelogs doorzoeken, compatibiliteit controleren en inschatten welke updates veilig kunnen worden doorgevoerd, wat in de praktijk veel tijd vraagt en vaak wordt uitgesteld (Behutiye e.a., 2024; Paschchenko e.a., 2018).

Onderzoek toont aan dat veel softwareprojecten hun dependencies niet systematisch bijhouden, waardoor een aanzienlijk deel verouderd raakt. Pashchenko et al. (Pashchenko e.a., 2018) stellen bijvoorbeeld dat een belangrijk percentage kwetsbare dependencies relatief eenvoudig opgelost kan worden door een update, maar dat dit in de praktijk niet gebeurt wegens tijdsdruk en gebrek aan overzicht.

Wanneer het updateproces niet systematisch gebeurt, stapelen onderhoudstaken zich op, waardoor het steeds moeilijker en tijdrovender wordt om de software up-to-date te houden. Dit fenomeen staat bekend als *technische schuld* (technical debt): de achterstand die ontstaat doordat noodzakelijke verbeteringen of updates te lang worden uitgesteld, wat leidt tot hogere onderhoudskosten en complexiteit op lange termijn (Behutiye e.a., 2024; Ruiz e.a., 2024). Empirisch onderzoek bevestigt dat technische schuld binnen bedrijven reële negatieve gevolgen heeft voor productiviteit en softwarekwaliteit (Chalmers University of Technology, 2020).

Veel organisaties gebruiken al geautomatiseerde dependencybots, zoals Dependabot of Renovate, die pull requests aanmaken zodra er nieuwe versies beschikbaar zijn (Erlenhov e.a., 2022). Deze oplossingen helpen om updates te detecteren, maar geven doorgaans weinig context over de inhoud en impact van de wijziging. Ontwikkelaars verliezen daardoor nog steeds tijd aan het interpreteren van changelogs, het inschatten van risico's en het beslissen of een pull request gemerged kan worden (Erlenhov e.a., 2022).

Binnen het bedrijf waar deze bachelorproef wordt uitgevoerd, leiden deze manuele analyses tot extra werkdruk voor een specifiek ontwikkelteam, namelijk het kleine kernteam dat verantwoordelijk is voor platform- en dependencybeheer over meerdere projecten en repositories heen.

Daardoor gaat er disproportioneel veel tijd naar het nakijken van dependency-updates, ten koste van andere ontwikkeltaken.

Deze bachelorproef richt zich daarom op het ontwikkelen van een AI-gedreven systeem dat bovenop bestaande tooling zoals Dependabot draait. Het systeem leest automatisch de door Dependabot aangemaakte pull requests in, analyseert de bijhorende changelogs en release notes, vat de essentie in begrijpelijke taal samen en ondersteunt het nemen van een beslissing (bijvoorbeeld mergen, uitstellen of extra acties uitvoeren). De tool moet onder andere kunnen controleren welke dependency wordt bijgewerkt, welk type update het betreft (patch, minor, major), wat de belangrijkste wijzigingen zijn en moet, indien gewenst, automatisch een pull request goedkeuren of sluiten volgens vooraf gedefinieerde regels.

Op die manier kan het updateproces slimmer, sneller en inzichtelijker worden gemaakt voor een duidelijk afgebakende doelgroep: het kleine kernteam dat binnen het bedrijf verantwoordelijk is voor dependencybeheer en platformonderhoud, in plaats van alle ontwikkelteams in het algemeen.

1.2. Onderzoeksvraag

Om het probleem van handmatig dependencybeheer en beperkte context bij updates op te lossen, wordt de volgende hoofdonderzoeksvraag geformuleerd:

> Hoe kan een AI-gedreven agent worden ingezet om dependency-updates uit bestaande tools voor automatisch dependencybeheer te analyseren en te beheren, zodat het kernteam dat verantwoordelijk is voor platform- en dependencybeheer efficiënter en met minder handmatig werk dependency-updates kan verwerken?

Hieruit vloeien de volgende deelvragen voort.

1.2.1. Deelvragen voor het beter begrijpen van het probleem

1. Wat zijn de belangrijkste uitdagingen en risico's bij het huidige, overwegend manuele dependency management voor het kernteam dat verantwoordelijk is voor platform- en dependencybeheer?
2. Welke bestaande tools en oplossingen voor automatisch dependencybeheer worden vandaag gebruikt, en welke sterke punten en beperkingen hebben deze oplossingen in de context van dit bedrijf?
3. Hoe leidt ophopende technische schuld op het vlak van dependency-updates ertoe dat er steeds meer tijd en inspanning nodig zijn voor het controleren en bijwerken van dependencies binnen het bedrijf?

1.2.2. Deelvragen richting de oplossing

4. Hoe kunnen AI-agents en workflow-automatiseringspl worden ingezet om informatie uit changelogs, release notes en dependency-pull-requests automatisch te interpreteren en in begrijpelijke vorm te communiceren naar het verantwoordelijke kernteam?
5. Welke functionele en niet-functionele vereisten moet een platform voor AI-gestuurd dependencybeheer vervullen, bijvoorbeeld op het vlak van integratie met versiebeheersystemen en registries, uitbreidbaarheid, beheer van regels en policies, auditlogging en performantie?
6. In welke mate voldoen geselecteerde AI-agent- of workflowplatformen aan deze vereisten, en hoe goed kunnen zij geïntegreerd worden in een proof-of-concept voor dependency management in de context van het bedrijf?

1.3. Onderzoeksdoelstelling

Het doel van dit onderzoek is om een proof-of-concept te ontwikkelen van een webapplicatie die automatisch dependency-updates detecteert, analyseert en communiceert via een geïntegreerde AI-agent, en om na te gaan in welke mate geselecteerde AI-agent- of workflowplatformen voldoen aan de vereisten van het bedrijf voor geautomatiseerd dependencybeheer. Eerder onderzoek toont aan dat AI-technieken succesvol kunnen worden ingezet binnen onderhoudstaken zoals changelog-analyse en code review, wat aantoont dat deze toepassing haalbaar en relevant is (Behutiye e.a., 2024; Joshi, 2025; Kim e.a., 2024; Zhu e.a., 2025).

Concreet zal de oplossing:

- per project de gebruikte dependencies bijhouden;
- automatisch nagaan of er nieuwe versies beschikbaar zijn via publieke registries (zoals npm, Maven Central, NuGet en PyPI) of via bestaande tools zoals Dependabot;
- changelogs, release notes en dependency-pull-requests analyseren met behulp van een AI-agent;
- het verantwoordelijke kernteam informeren over de aard en impact van de updates, inclusief een korte, begrijpelijke samenvatting en een indicatie of de update veilig lijkt om door te voeren;
- regels implementeren om, waar mogelijk, automatisch een pull request goed te keuren of te sluiten, zodat het kernteam minder manuele beslissingen hoeft te nemen;

- geselecteerde platformen beoordelen op geschiktheid op basis van gebruiksgemak, integratiemogelijkheden, onderhoudbaarheid, performantie en mate waarin zij voldoen aan de gedefinieerde vereisten.

De tool zal opgebouwd worden met `Next.js` als frontend, een C# (.NET) of Java (Spring Boot) backend, en een database (PostgreSQL, MongoDB of Supabase) voor gebruikers- en projectbeheer. Het onderzoek resulteert in:

- een prototype (demo) dat de werking van het systeem aantoont binnen een of enkele projecten van het bedrijf;
- een evaluatie van de toegevoegde waarde van AI bij dependency management voor het specifieke kernteam;
- een aanbeveling welk type platform, en eventueel welke concrete technologie, het meest geschikt is voor verdere toepassing binnen het bedrijf.

1.4. Opzet van deze bachelorproef

De bachelorproef volgt een klassieke onderzoeksstructuur en is als volgt opgebouwd:

- **Hoofdstuk 1 – Inleiding en probleemstelling:** Dit hoofdstuk beschrijft de context, de probleemstelling, de onderzoeksvragen en de doelstellingen van de bachelorproef.
- **Hoofdstuk 2 – Literatuurstudie:** In dit hoofdstuk wordt de relevante literatuur rond dependency management, bestaande tools en oplossingen, technische schuld en AI-agent-frameworks en workflow-automatisering geanalyseerd.
- **Hoofdstuk 3 – Methodologie en implementatie:** Dit hoofdstuk bespreekt de onderzoeksaanpak, het ontwerp van de systeemarchitectuur, de selectie van platformen en de implementatie van de proof-of-concept.
- **Hoofdstuk 4 – Verwachte resultaten:** In dit hoofdstuk worden de verwachte resultaten van de implementatie en experimenten gepresenteerd.
- **Hoofdstuk 5 – Verwachte conclusie:** Dit hoofdstuk formuleert de verwachte belangrijkste bevindingen op basis van de onderzoeksvragen

2. Literatuurstudie

Hier beschrijf je de *state-of-the-art* rondom je gekozen onderzoeksdomein, d.w.z. een inleidende, doorlopende tekst over het onderzoeksdomein van je bachelorproef. Je steunt daarbij heel sterk op de professionele *vakliteratuur*, en niet zozeer op populariserende teksten voor een breed publiek. Wat is de huidige stand van zaken in dit domein, en wat zijn nog eventuele open vragen (die misschien de aanleiding waren tot je onderzoeksvraag!)?

Je mag de titel van deze sectie ook aanpassen (literatuurstudie, stand van zaken, enz.). Zijn er al gelijkaardige onderzoeken gevoerd? Wat concluderen ze? Wat is het verschil met jouw onderzoek?

Verwijs bij elke introductie van een term of bewering over het domein naar de vakliteratuur, bijvoorbeeld (**Hykes2013**)! Denk zeker goed na welke werken je refereert en waarom.

Draag zorg voor correcte literatuurverwijzingen! Een bronvermelding hoort thuis *binnen* de zin waar je je op die bron baseert, dus niet er buiten! Maak meteen een verwijzing als je gebruik maakt van een bron. Doe dit dus *niet* aan het einde van een lange paragraaf. Baseer nooit teveel aansluitende tekst op eenzelfde bron.

Als je informatie over bronnen verzamelt in JabRef, zorg er dan voor dat alle nodige info aanwezig is om de bron terug te vinden (zoals uitvoerig besproken in de lessen Research Methods).

Je mag deze sectie nog verder onderverdelen in subsecties als dit de structuur van de tekst kan verduidelijken.

3. Methodologie

Hier beschrijf je hoe je van plan bent het onderzoek te voeren. Welke onderzoekstechniek ga je toepassen om elk van je onderzoeksvragen te beantwoorden? Gebruik je hiervoor literatuurstudie, interviews met belanghebbenden (bv. voor requirements-analyse), experimenten, simulaties, vergelijkende studie, risico-analyse, PoC, ...?

Valt je onderwerp onder één van de typische soorten bachelorproeven die besproken zijn in de lessen Research Methods (bv. vergelijkende studie of risico-analyse)? Zorg er dan ook voor dat we duidelijk de verschillende stappen terug vinden die we verwachten in dit soort onderzoek!

Vermijd onderzoekstechnieken die geen objectieve, meetbare resultaten kunnen opleveren. Enquêtes, bijvoorbeeld, zijn voor een bachelorproef informatica meestal **niet geschikt**. De antwoorden zijn eerder meningen dan feiten en in de praktijk blijkt het ook bijzonder moeilijk om voldoende respondenten te vinden. Studenten die een enquête willen voeren, hebben meestal ook geen goede definitie van de populatie, waardoor ook niet kan aangetoond worden dat eventuele resultaten representatief zijn.

Uit dit onderdeel moet duidelijk naar voor komen dat je bachelorproef ook technisch voldoende diepgang zal bevatten. Het zou niet kloppen als een bachelorproef informatica ook door bv.

een student marketing zou kunnen uitgevoerd worden.

Je beschrijft ook al welke tools (hardware, software, diensten, ...) je denkt hiervoor te gebruiken of te ontwikkelen.

Probeer ook een tijdschatting te maken. Hoe lang zal je met elke fase van je onderzoek bezig zijn en wat zijn de concrete *deliverables* in elke fase?

4. Verwacht resultaat, conclusie

Hier beschrijf je welke resultaten je verwacht. Als je metingen en simulaties uitvoert, kan je hier al mock-ups maken van de grafieken samen met de verwachte conclusies. Benoem zeker al je assen en de onderdelen van de grafiek die je gaat gebruiken. Dit zorgt ervoor dat je concreet weet welk soort data je moet verzamelen en hoe je die moet meten.

Wat heeft de doelgroep van je onderzoek aan het resultaat? Op welke manier zorgt jouw bachelorproef voor een meerwaarde?

Hier beschrijf je wat je verwacht uit je onderzoek, met de motivatie waarom. Het is **niet** erg indien uit je onderzoek andere resultaten en conclusies vloeien dan dat je hier beschrijft: het is dan juist interessant om te onderzoeken waarom jouw hypothesen niet overeenkomen met de resultaten.

Referenties

Behutiye, W. N., Rodriguez, P., Oivo, M., & Tosun, A. (2024). Analyzing the Concept of Technical Debt in the Context of Agile Software Development: A Systematic Literature Review. *arXiv*. https://arxiv.org/abs/2401.14882

Chalmers University of Technology. (2020). Technical Debt: An Empirical Investigation of Its Harmfulness and Management Strategies in Industry. https://research.chalmers.se/en/publication/518318

Erlenhov, L., De Oliveira Neto, F. G., & Leitner, P. (2022). Dependency Management Bots in Open-Source Systems—Prevalence and Adoption. *PeerJ Computer Science*, *8*, e849. https://doi.org/10.7717/peerj-cs.849

Joshi, S. (2025). Review of Autonomous Systems and Collaborative AI Agent Frameworks. *International Journal of Science and Research Archive*, *14*(02), 961–972. https://doi.org/10.30574/ijsra.2025.14.2.0439

Kim, Y., Abdelaziz, A. E., Castro Ferreira, T., Al-Badrashiny, M., & Sawaf, H. (2024). Bel Esprit: Multi-Agent Framework for Building AI Model Pipelines. *arXiv*. https://doi.org/10.48550/arXiv.2412.14684

Pashchenko, I., Plate, H., Ponta, S. E., Sabetta, A., & Massacci, F. (2018). Vulnerable Open Source Dependencies: Counting Those That Matter. *Proceedings of the 12th International Symposium on Empirical Software Engineering and Measurement (ESEM)*. https://arxiv.org/abs/1808.09753

Ruiz, J., Aguilar, J., Garcilazo, J., & Aguileta, A. (2024). A Tertiary Study on Technical Debt Management Over the Last Lustrum. *International Journal of Computers and Their Applications*, *31*(1). https://ijcopi.org/ojs/article/view/577

Zhu, H., Qin, T., Zhu, K., Huang, H., Guan, Y., Xia, J., & Liu, J. (2025). OAgents: An Empirical Study of Building Effective Agents. *arXiv*. https://doi.org/10.48550/arXiv.2506.15741