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: B. Pattyn (Springbok Agency, bert.pattyn@springbokagency.com)

Samenvatting

Deze bachelorproef onderzoekt hoe een AI-gedreven agent kan worden ingezet om softwaredependencies automatisch te analyseren en te beheren in combinatie met een dependencybot. Vanuit de vastgestelde problematiek — veel manueel werk, verhoogde technische schuld en beperkte context bij dependency-updates — wordt een proof-of-concept webapplicatie ontwikkeld die pull requests, changelogs en release notes automatisch uitleest, samenvat en van beslissingsvoorstellen voorziet voor een klein kernteam dat platform- en dependencybeheer opneemt.
De oplossing wordt opgebouwd met een moderne webstack (Next.js, .NET/Java en een relationele of NoSQL-database) en integreert met publieke package registries en een eigen dependencybot om updates te detecteren. Via een vergelijkende studie van geselecteerde AI-agent- en workflowplatformen wordt onderzocht in welke mate deze voldoen aan gedefinieerde functionele en niet-functionele vereisten, zoals integratiemogelijkheden, onderhoudbaarheid en performantie. Verwacht wordt dat de aanpak de manuele beoordelingslast merkbaar vermindert, de zichtbaarheid van technische schuld rond dependencies verhoogt en een herhaalbaar proces oplevert dat als basis kan dienen voor verdere uitrol binnen de organisatie.

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; Pashchenko 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[1] of Renovate[2], 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[3] 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 automatiseringsplatformen 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?

[1] Dependabot is de ingebouwde GitHub-tool voor het automatisch detecteren van kwetsbare of verouderde dependencies en het aanmaken van update-pull-requests.(GitHub, z.d.)

[2] Renovate is een open-source dependencybot die geautomatiseerde update-pull-requests en uitgebreide configuratiemogelijkheden biedt voor diverse platformen en package managers.(Mend, z.d.)

[3] Zie (GitHub, z.d.) voor een overzicht van de functionaliteit, configuratieopties en best practices van Dependabot.

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[4], Maven Central[5], NuGet[6] en PyPI[7]) of via bestaande tools zoals Dependabot[8];
- 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`[9] als frontend, een C#[10] (.NET) of Java (Spring Boot[11]) backend, en een database (PostgreSQL[12], MongoDB[13] of Supabase[14]) 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.

[4] npm is het standaardpakketbeheerplatform voor het JavaScript-ecosysteem.(npm, Inc., z.d.)

[5] Maven Central is de centrale repository voor Java-artifacts in het Maven-ecosysteem.(Sonatype, z.d.)

[6] NuGet is de officiële package manager en online galerij voor .NET-bibliotheken.(Microsoft, z.d.-b)

[7] PyPI is de officiële Python Package Index voor het publiceren en distribueren van Python-pakketten.(Python Software Foundation, z.d.)

[8] Zie opnieuw (GitHub, z.d.) voor de integratie van Dependabot met verschillende ecosystemen.

[9] Next.js is een React-gebaseerd webframework voor het bouwen van full-stack webapplicaties met server-side rendering en API-routes.(Vercel, z.d.)

[10] C# is een moderne, objectgeoriënteerde programmeertaal op het .NET-platform.(Microsoft, z.d.-a)

[11] Spring Boot is een Java-framework dat het bouwen en configureren van productieklare microservices vereenvoudigt.(VMware, z.d.)

[12] PostgreSQL is een open-source relationeel databasemanagementsysteem dat sterk inzet op betrouwbaarheid en uitbreidbaarheid.(The PostgreSQL Global Development Group, 2025)

[13] MongoDB is een documentgeoriënteerde NoSQL-database gericht op flexibiliteit en schaalbaarheid.(MongoDB Inc., z.d.)

[14] Supabase is een ontwikkelplatform bovenop PostgreSQL dat out-of-the-box API's, authenticatie en opslag aanbiedt.(Supabase, z.d.)

- **Hoofdstuk 5 – Verwachte conclusie:** Dit hoofdstuk formuleert de verwachte belangrijkste bevindingen op basis van de onderzoeksvragen.

2. Literatuurstudie

2.1. Dependency management en kwetsbare dependencies

Moderne softwareprojecten steunen sterk op open-source dependencies, waardoor kwetsbaarheden in externe bibliotheken rechtstreeks een impact kunnen hebben op de veiligheid en betrouwbaarheid van toepassingen.(Pashchenko e.a., 2018) Pashchenko et al. tonen aan dat een groot deel van de aangetroffen kwetsbare dependencies in de praktijk relatief eenvoudig verholpen kan worden door te upgraden naar een veiligere versie, maar dat ontwikkelteams deze updates vaak niet consequent doorvoeren.(Pashchenko e.a., 2018)

In hun proefondervindelijk onderzoek onderscheiden Pashchenko et al. tussen daadwerkelijk gedeployde kwetsbare dependencies en libraries die enkel tijdens ontwikkeling of testing gebruikt worden, en stellen zij vast dat het merendeel van de reëel kwetsbare dependencies door de eigen ontwikkelaars of directe onderhouders kan worden opgelost.(Pashchenko e.a., 2018) Dit onderstreept het belang van systematisch dependencybeheer en duidelijke inzichten in welke dependencies effectief risico vormen in productie omgevingen.(Pashchenko e.a., 2018)

2.2. Technische schuld en onderhoudslast

Het uitstellen van dependency-updates draagt bij aan de opbouw van technische schuld, waardoor onderhoud en toekomstige wijzigingen steeds meer inspanning vereisen.(Behutiye e.a., 2024; Ruiz e.a., 2024) Behutiye et al. analyseren de rol van technische schuld in agile omgevingen en beschrijven hoe keuzes die op korte termijn tijdswinst opleveren, op langere termijn leiden tot hogere kosten en complexere wijzigingen.(Behutiye e.a., 2024)

Ruiz et al. bespreken in hun tertiaire studie dat technische-schuldmanagement de voorbije jaren een groeiend onderzoeksdomein is geworden, waarbij organisaties worstelen met het in kaart brengen, prioriteren en terugdringen van opgebouwde schuld.(Ruiz e.a., 2024) Empirisch werk van Chalmers University of Technology bevestigt dat technische schuld niet louter een theoretisch concept is, maar in industriële context concreet voelbaar is in de vorm van verminderde productiviteit, hogere foutkans en vertragingen in softwarelevering.(Chalmers University of Technology, 2020)

2.3. Dependencybots en hun beperkingen

Om de handmatige last rond dependencybeheer te verlagen, zetten veel projecten dependency management bots in, zoals Dependabot[15] en Renovate[16].(Erlenhov e.a., 2022) Erlenhov et al. voeren een kwantitatieve en kwalitatieve studie uit naar dependencybots in open-source systemen en stellen vast dat slechts een beperkt aantal van de geanalyseerde geautomatiseerde tools voldoet aan de criteria van volwaardige “Devbots”, die autonoom bijdragen aan de codebasis.(Erlenhov e.a., 2022)

Hun resultaten tonen dat projecten vaak experimenteren met meerdere bots en regelmatig wisselen, onder meer door problemen met ruis (te veel pull requests), beperkte configureerbaarheid en moeilijkheden om de impact van voorgestelde updates goed te begrijpen.(Erlenhov e.a., 2022) Hoewel dependencybots het detecteren en aanmaken van update-pull-requests automatiseren, blijft de inhoudelijke beoordeling van changelogs, compatibiliteit en risico’s grotendeels bij ontwikkelaars liggen, wat aansluit bij de probleemstelling van deze bachelorproef.(Erlenhov e.a., 2022; Pashchenko e.a., 2018)

2.4. AI, agents en workflow automatisering

Recente literatuur verkent hoe AI-agents en multi-agentframeworks kunnen worden ingezet om complexe taken in softwareontwikkeling en data-intensieve workflows te coördineren.(Joshi, 2025; Kim e.a., 2024; Zhu e.a., 2025) Joshi biedt een overzicht van autonome systemen en collaboratieve AI-agentframeworks en benadrukt dat zulke agents vooral nuttig zijn in scenario’s met veel herhalende beslissingen op basis van tekstuele en gestructureerde input, zoals logs, notificaties en rapporten.(Joshi, 2025)

Kim et al. introduceren met Bel Esprit een multi-agentframework voor het opbouwen van AI-modelpijplijnen, waarbij verschillende gespecialiseerde agents samenwerken om complexere taken modulair af te handelen.(Kim e.a., 2024) Zhu et al. bestuderen in OAgents empirisch welke ontwerpkeuzes leiden tot effectieve agents, en bespreken onder meer het belang van duidelijke taakafbakening, robuuste toolintegratie en feedbackloops voor betrouwbare beslissingsondersteuning.(Zhu e.a., 2025) Deze inzichten zijn relevant voor het ontwerp van een AI-agent die dependency-updates

[15] Dependabot is de ingebouwde GitHub-tool voor het monitoren van kwetsbaarheden en het automatisch aanmaken van update-pull-requests voor ondersteunde ecosystemen.(GitHub, z.d.)

[16] Renovate is een open-source dependencybot die automatisch pull requests genereert voor versie- en beveiligingsupdates, met uitgebreide configuratiemogelijkheden en ondersteuning voor meerdere platformen.(Mend, z.d.)

interpreteert, samenvat en beslissingsvoorstellen formuleert voor een onderhoudsteam.

Naast generieke agentframeworks beschrijft Wali et al. hoe workflow-automatisering met n8n[17] operationele efficiëntie kan verhogen door repetitieve, gegevensgedreven processen te coördineren over verschillende systemen heen.(Wali e.a., 2025) Hoewel hun casus zich richt op een financiële context, illustreert de studie dat low-code workflowtools geschikt zijn om integraties met externe APIs, databronnen en notificatiekanalen te realiseren, wat ook essentieel is voor een geautomatiseerde dependency-updatepipeline.(Wali e.a., 2025)

2.5. Onderzoeksniche: AI-ondersteund dependencybeheer

Samengenomen schetst de literatuur een duidelijk beeld: kwetsbare of verouderde dependencies vormen een reëel risico, maar kunnen vaak verholpen worden door updates die vandaag onvoldoende systematisch worden uitgevoerd.(Behutiye e.a., 2024; Pashchenko e.a., 2018) Tegelijk tonen studies naar technische schuld dat het structureel uitstellen van onderhoud, waaronder dependency updates, leidt tot een groeiende onderhoudslast en achterstand.(Behutiye e.a., 2024; Chalmers University of Technology, 2020; Ruiz e.a., 2024)

Onderzoek naar dependencybots maakt duidelijk dat huidige oplossingen weliswaar updates detecteren en pull requests genereren, maar ontwikkelaars nog steeds belasten met het interpreteren van changelogs en het beoordelen van risico's en prioriteiten.(Erlenhov e.a., 2022) De recente vooruitgang in AI-agents en workflow-automatisering suggereert dat een volgende stap mogelijk is: een systeem waarin een AI-agent niet alleen updates signaleert, maar ook de inhoud van release notes en changelogs analyseert, samenvat en contextafhankelijk advies geeft aan een klein kernteam dat verantwoordelijk is voor dependencybeheer.(Joshi, 2025; Kim e.a., 2024; Wali e.a., 2025; Zhu e.a., 2025)

Deze bachelorproef positioneert zich precies in deze niche door een AI-gestuurde laag bovenop bestaande dependencybots te onderzoeken en te prototypen, met de expliciete focus op het verlagen van de manuele beoordelingslast en het beheersbaar houden van technische schuld rond dependencies in een bedrijfscontext.

3. Methodologie

Dit onderzoek volgt hoofdzakelijk de opzet van een vergelijkende studie met proof-of-concept, aangevuld met literatuuronderzoek en een beperkte requirements-analyse bij een bedrijfscontext. Het doel is om enerzijds het probleem en de oplossingsruimte rond dependencybeheer in kaart te brengen, en anderzijds een prototype te bouwen en technisch te evalueren binnen een realistische ontwikkelomgeving.

3.1. Onderzoeksaanpak per onderzoeksvraag

Om de eerste groep deelvragen, die gericht zijn op het beter begrijpen van het probleem, te beantwoorden, wordt een systematische literatuurstudie uitgevoerd rond dependency management, kwetsbare dependencies, technische schuld en bestaande dependencybots. Hierbij wordt voornamelijk gebruikgemaakt van wetenschappelijke artikels, technische rapporten en documentatie van industriële tools.

Voor de deelvragen die gericht zijn op het definiëren van de oplossing, wordt een combinatie van requirements-analyse, proof-of-concept-ontwikkeling en technische experimenten ingezet. De requirements-analyse gebeurt via gesprekken met de betrokken medewerkers binnen het bedrijf (vooral leden van het kernteam dat platform- en dependencybeheer opneemt), gericht op het expliciteren van:

- functionele eisen (detectie van updates, analyse en samenvatting van changelogs en release notes).;
- niet-functionele eisen (performantie, integratiemogelijkheden, onderhoudbaarheid, beheer van regels en policies, auditlogging).

De resultaten van deze analyse vormen de basis voor een set concrete evaluatiecriteria die later gebruikt worden om de geselecteerde AI-agent- en workflowplatformen te beoordelen.

3.2. Proof-of-concept en technische implementatie

De kern van het onderzoek bestaat uit het bouwen van een proof-of-concept webapplicatie die:

- afhankelijkheden per project bijhoudt en koppelt aan informatie uit publieke registries;
- inkomende dependency-updates inleest en de bijbehorende changelogs en release notes verwerkt;
- via een AI-agent tekstuele samenvattingen en beslissingsvoorstellen genereert;
- het kernteam een overzicht biedt van openstaande updates, prioriteiten en voorgestelde acties.

[17] n8n is een open-source, low-code workflow-automatiseringstool waarmee integraties tussen API's, databronnen en notificatiesystemen visueel gemodelleerd en uitgevoerd kunnen worden.(Wali e.a., 2025)

Technisch wordt de proof-of-concept gerealiseerd met:

- een `Next.js`-frontend voor de gebruikersinterface;
- een backend in C# (.NET) of Java (Spring Boot), afhankelijk van de voorkeuren en standaardtechnologieën binnen het bedrijf;
- een database-oplossing (PostgreSQL, MongoDB of Supabase) voor opslag van gebruikers, projecten, dependency-informatie en logging.

De AI-agentlaag wordt eerst in een basisopzet gerealiseerd (bijvoorbeeld één gekozen platform of framework), waarbij de focus ligt op het correct interpreteren en samenvatten van changelogs en release notes. Daarna wordt onderzocht in welke mate alternatieve agent- of workflowplatformen kunnen worden ingezet om dezelfde functionaliteit te ondersteunen, zodat een zinvolle vergelijking mogelijk is.

3.3. Vergelijkende evaluatie van platformen

Om de onderzoeksvraag rond de geschiktheid van verschillende AI-agent- en workflowplatformen te beantwoorden, wordt een vergelijkende evaluatie uitgewerkt. Op basis van de eerder gedefinieerde vereisten worden meetbare criteria opgesteld, zoals:

- integratiemogelijkheden met Git-repositories en externe APIs;
- complexiteit van configuratie en ontwikkelervaring (bijvoorbeeld aantal stappen om een eerste workflow op te zetten);
- onderhoudbaarheid (structuur van workflows, versiebeheer, herbruikbaarheid van componenten);
- performantie-indicatoren, zoals doorlooptijd voor het verwerken van een set voorbeelden.

Er wordt een reeks gestandaardiseerde scenario's gedefinieerd (bijvoorbeeld een set representatieve dependency-updates met verschillende types wijzigingen en risico's). Voor elk geselecteerd platform wordt dezelfde set scenario's geïmplementeerd en uitgevoerd, zodat de resultaten langs de gedefinieerde criteria objectief kunnen worden vergeleken. De uitkomsten van deze experimenten worden gebruikt om de sterktes en zwaktes van elk platform in de context van dit specifieke probleemdomein te identificeren.

3.4. Tools en middelen

Voor de praktische uitvoering van het onderzoek wordt gebruikgemaakt van:

- versiebeheer (bijvoorbeeld GitHub) voor de codebasis en configuratiebestanden;
- een zelf gemaakte dependencybot als bron van automatische updates;
- publieke registries (npm, Maven Central, NuGet, PyPI) voor versie-informatie en metadata over dependencies;
- gekozen AI-agent- of workflowplatformen voor de implementatie van de AI-gedreven analysetaken;
- standaardontwikkeltools (IDE, containerisatie, CI/CD) voor het bouwen, testen en uitrollen van de proof-of-concept.

3.5. Fasering en tijdsplanning

Het onderzoek wordt opgedeeld in opeenvolgende fasen met bijhorende deliverables, zoals weergegeven in Figuur 1.

- **Fase 1 – Oriëntatie en literatuurstudie (reeds lopend):** in deze fase worden de probleemstelling verfijnd, relevante bronnen verzameld en geanalyseerd, en potentiële co-promotoren en bedrijfscontexten verkend. Deliverables: onderzoeksvoorstel, voorlopige literatuurstudie, bevestiging of afwijzing van co-promotor(en).
- **Fase 2 – Requirements-analyse en architectuurontwerp:** gesprekken met het betrokken bedrijf om de concrete requirements en context te verduidelijken, uitwerking van use-cases en systeemarchitectuur van de proof-of-concept. Deliverables: requirementsdocument, architectuurschets, lijst van evaluatiecriteria voor de platformen.
- **Fase 3 – Implementatie van de proof-of-concept:** ontwikkeling van frontend, backend en database, integratie met dependencybot en registries, en eerste versie van de AI-agent voor changelog- en release note-analyse. Deliverables: werkend prototype, technische documentatie, testcases en voorbeelddata (updates, changelogs).
- **Fase 4 – Vergelijkende evaluatie en experimenten:** implementatie van de gestandaardiseerde scenario's op de geselecteerde platformen, uitvoeren van metingen (bijvoorbeeld doorlooptijd, configuratie-inspanning) en analyse van de resultaten. Deliverables: experimenteel protocol, ruwe meetdata, geanalyseerde resultaten en tabellen/grafieken.
- **Fase 5 – Rapportering en verwachte conclusie:** uitwerken van de uiteindelijke bachelorproef, formuleren van conclusies en aanbevelingen op basis van de resultaten, en reflectie op eventuele verschillen tussen de

verwachte en daadwerkelijke uitkomsten. De-liverables: volledige bachelorproef, presentatie en demonstratiemateriaal.

4. Verwacht resultaat

Op basis van de probleemstelling en onderzoeksdoelstelling wordt verwacht dat het ontwikkelde proof-of-concept in staat zal zijn om dependency updates automatisch te detecteren, te analyseren en in een compacte, begrijpelijke vorm te communiceren naar het verantwoordelijke kernteam. Concreet wordt verwacht dat de AI-agent changelogs, release notes kan omzetten naar heldere samenvattingen met een indicatie van het type wijziging (beveiligingsfix, bugfix, nieuwe functionaliteit) en de vermoedelijke impact op het project.

Verder wordt verwacht dat het systeem de werklast voor het kernteam merkbaar reduceert door een deel van de beslissingen rond “veilige” updates te automatiseren. Denk daarbij aan scenario’s waarin kleine, backward compatible updates te updates (bijvoordbeeld patch- of beperkte minor-releases) automatisch kunnen worden goedgekeurd of voorbereid voor merge, terwijl potentieel risicovolle major-updates expliciet gemarkeerd worden voor manuele review. Indien er voldoende data verzameld kan worden uit tests of logbestanden, worden bovendien verwacht dat er minimaal een eerste, kwantitatieve inschatting kan worden gemaakt van de tijdswinst (bijvoorbeeld aantal manuele reviews per maand, gemiddelde doorlooptijd per update) in vergelijking met de situatie voor de invoering van de tool.

De doelgroep van dit onderzoek, namelijk het kleine kernteam dat binnen het bedrijf verantwoordelijk is voor dependencybeheer en platformonderhoud, haalt hieruit meerwaarde doordat zij:

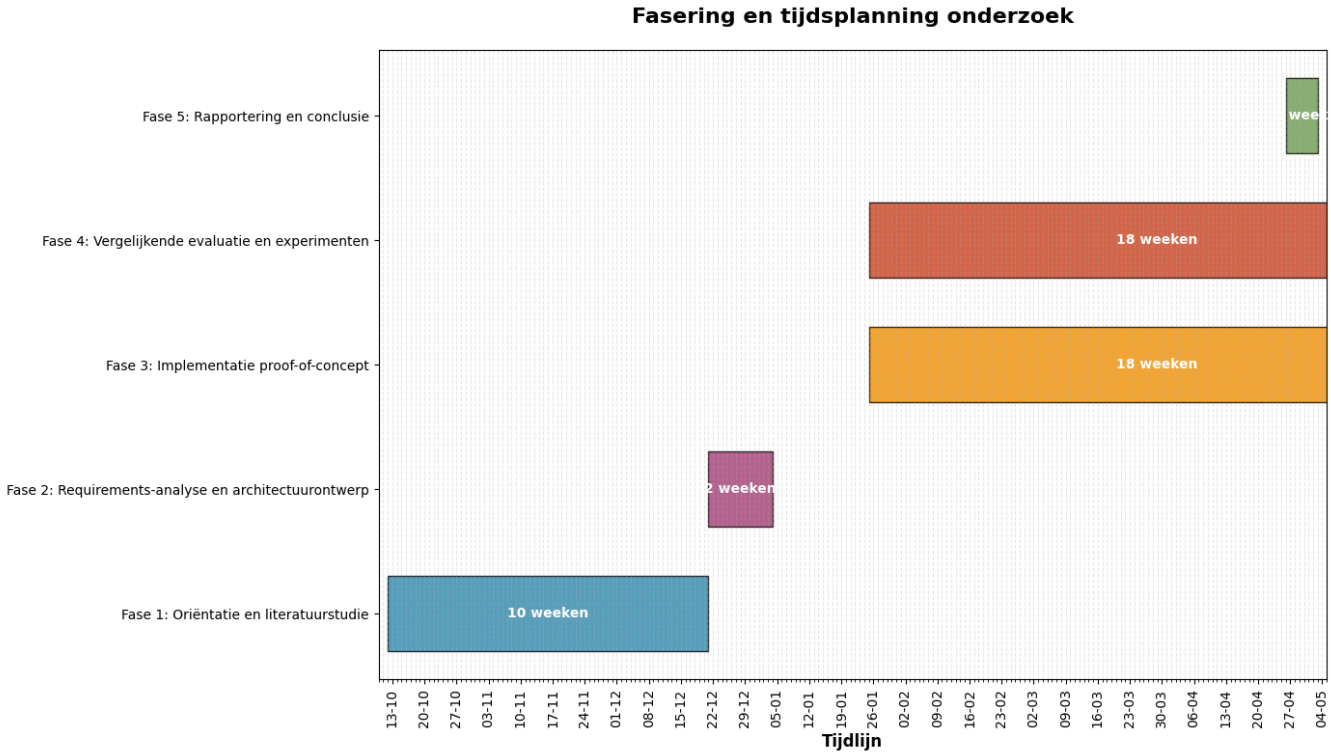
- sneller kunnen bepalen welke updates prioriteit hebben en welke zonder veel risico kunnen worden doorgevoerd;
- minder tijd verliezen aan het doorpluizen van changelogs en release notes;
- een beter overzicht krijgen van de actuele staat van dependencies en de bijhorende technische schuld;
- een herhaalbaar proces in handen krijgen dat later opgeschaald kan worden naar andere teams of projecten binnen de organisatie.

5. Verwacht conclusie

Op basis van de literatuur en de beoogde proof-of-concept wordt verwacht dat de bachelorproef zal aantonen dat een AI-gestuurde laag de manuele beoordelinglast rond dependency-updates kan verlagen, zonder daarbij de controle volledig uit handen te geven. Verwacht wordt dat de resultaten zullen aangeven dat het combineren van automatische detectie met een AI-agent voor interpretatie en samenvatting een efficiënter en transparanter updateproces oplevert voor het kernteam.

Verder wordt verwacht dat uit de evaluatie van de geselecteerde AI-agent of workflowplatformen zal blijken dat niet een enkel platform op alle criteria domineert, maar dat er duidelijke verschillen zijn op het vlak van integratiemogelijkheden, gebruiksgemak, onderhoudbaarheid en performantie. De verwachte conclusie is dat op basis van de gedefinieerde vereisten een onderbouwde aanbeveling kan worden geformuleerd voor het meest geschikte platform binnen de context van het betrokken bedrijf, en dat de inzet van een AI-agent een zinvolle stap is in het beperken van technische schuld rond dependencies. Indien de uiteindelijke resultaten afwijken van deze verwachtingen, biedt dit een interessante basis om te analyseren welke aanames niet klopten en welke randvoorwaarden cruciaal zijn voor het succes van AI-ondersteund dependencybeheer.

6. Figuren



Figuur 1: Fasering en tijdsplanning van het onderzoek

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

GitHub. (z.d.). *Keeping Your Supply Chain Secure with Dependabot* [Geraadpleegd op 12 december 2025]. https://docs.github.com/en/code-security/dependabot

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

Mend. (z.d.). *Renovate Documentation* [Geraadpleegd op 12 december 2025]. https://docs.renovatebot.com/

Microsoft. (z.d.-a). *C# – A Modern, Open-Source Programming Language* [Geraadpleegd op 12 december 2025]. https://dotnet.microsoft.com/en-us/languages/csharp

Microsoft. (z.d.-b). *NuGet Gallery | Home* [Geraadpleegd op 12 december 2025]. https://www.nuget.org/

MongoDB Inc. (z.d.). *MongoDB: The World's Leading Modern Database* [Geraadpleegd op 12 december 2025]. https://www.mongodb.com/

npm, Inc. (z.d.). *npm | Home* [Geraadpleegd op 12 december 2025]. https://www.npmjs.com/

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

Python Software Foundation. (z.d.). *PyPI: The Python Package Index* [Geraadpleegd op 12 december 2025]. https://pypi.org/

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

Sonatype. (z.d.). *Maven Central* [Geraadpleegd op 12 december 2025]. https://central.sonatype.com/

Supabase. (z.d.). *Supabase: The Postgres Development Platform* [Geraadpleegd op 12 december 2025]. https://supabase.com/

The PostgreSQL Global Development Group. (2025). *PostgreSQL* [Geraadpleegd op 12 december 2025]. https://www.postgresql.org/

Vercel. (z.d.). *Next.js by Vercel – The React Framework* [Geraadpleegd op 12 december 2025]. https://nextjs.org/

VMware. (z.d.). *Spring Boot* [Geraadpleegd op 12 december 2025]. https://spring.io/projects/spring-boot

Wali, M., Nasir, N., & Iqbal, T. (2025). Implementing Workflow Automation with n8n to Enhance Operational Efficiency and Performance in the Sharia Cooperative of Bank Indonesia, Aceh Province. *Journal Digital Technology Trend*, *4*(1), 36–47. https://doi.org/10.56347/jdtt.v4i1.341

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