ONDERZOEKSVOORSTEL

Proof of concept: De update automatiseren van Angular versie 16 naar Angular versie 20 in de applicaties van een end-to-end kredietdienstverlener.

Bachelorproef, 2025-2026

Wauters Sander

E-mail: sander.wauters@student.hogent.be

Samenvatting

Updaten naar een nieuwe versie van een softwareframework kan veel tijd in beslag nemen, zeker als de huidige applicatie meerdere versies achterloopt. Het doel van dit onderzoek is om de update van het Angular webframework van versie 16 naar versie 20 te automatiseren, bij meerdere applicaties, met als doel de onderhoudstijd voor de developers te verlagen. Framework updates kunnen de performantie en veiligheid van de applicatie verbeteren. Het is daarom van belang deze tijdig uit te voeren. Deze updates zijn niet altijd even simpel om toe te passen, zeker als deze gepaard gaan met veranderingen aan de broncode van de applicatie. In grotere broncodes neemt het proces van systematisch alle veranderingen aan te brengen veel tijd in beslag. Dit onderzoek begint met een oplijsting te maken van alle nodige aanpassingen tussen Angular versie 16 en versie 20. Vervolgens worden de verschillende manieren om aanpassingen automatisch uit te voeren onderzocht. Op basis hiervan wordt een applicatie uitgewerkt die het updateproces zal automatiseren. Deze applicatie wordt eerst getest in een gecontroleerde omgeving en vervolgens op een reële applicatie van een bedrijf. De effectiviteit wordt beoordeeld aan de hand van het aantal veranderingen dat de applicatie correct kan uitvoeren ten opzichte van het totaal aantal uit te voeren aanpassingen. Er wordt verwacht dat 65% van alle nodige veranderingen automatisch uitgevoerd kan worden. Het automatisatieproces zal naar verwachting de nodige tijd voor de applicaties te updaten verminderen.

Keuzerichting: Mobile & Enterprise development
Sleutelwoorden: Angular, Static code analysis, automatisatie

Inhoudsopgave

1 Inleiding 1
2 Literatuurstudie 2
2.1 Uit te voeren veranderingen 2
2.2 Automatisatie process 2
3 Methodologie 2
3.1 Literatuurstudie 2
3.2 Ontwikkeling van de updater 3
3.3 Evaluatie 3
4 Verwacht resultaat, conclusie 3
Referenties 3

1. Inleiding

Het bedrijf Stater is een end-to-end dienstverlener voor zowel hypothecaire als consumentenkrediete. Ze ondersteunen de kredietverstrekker voor de dienstverlening aan consumenten. Binnen het bedrijf zijn er verschillende applicaties die gebruikmaken van het Angular framework. Angular is een open-source front-end framework gebaseerd op de TypeScript programmeertaal voor de ontwikkeling van dynamische webapplicaties (Cincovic e.a., 2019). Momenteel is Angular versie 20 (v.20) de meest recente stabiele versie. Binnen Stater maken de applicaties gebruik van Angular versie 16 (v.16); het bedrijf is van plan de applicaties te updaten naar de recentste versie, Angular v.20.

Volgens de studie door Vaniea en Rashidi (2016) zijn software-updates nodig omdat het van nieuwe functionaliteiten introduceert, de performantie van de applicatie kan verbeteren en verzekert dat de software compatibel blijft met nieuwe software en hardware. Verder omschrijft deze bron dat het up-to-date houden van software cruciaal is om de cyberveiligheid te garanderen.

Het grote verschil in versies zorgt ervoor dat het updaten van alle applicaties veel tijd in beslag neemt. Dit is geen eenmalig probleem; volgens Callaghan (2023) krijgt Angular een nieuwe versie om de 6 maanden. Volgens Kaur en Singh (2015) neemt het onderhoud van een softwareproject gemiddeld 60% van de kostprijs in beslag. Een manier om de tijd voor software-onderhoud in te korten is daarom best interessant. Hieruit komt de vraag: In welke mate kan de automatisering van het updateproces van Angular v.16 naar v.20, bij meerdere applicaties, de onderhoudstijd voor developers verlagen? Om deze vraag te beantwoorden zijn de volgende deelvragen geformuleerd:

- Hoeveel veranderingen moeten uitgevoerd worden om Angular van v.16 naar v.20 te up-

daten?

- Wat zijn statistiek gezien de meest voorkomende problemen bij het refactoren van code?
- Welke manieren bestaan om code automatisch aan te passen zonder ongewenste veranderingen uit te voeren?
- Welke manier om code automatisch aan te passen is het meest geschikt om in deze casus toe te passen?

Gedurende dit onderzoek zal een applicatie ontwikkeld worden die een softwareproject in Angular v.16 automatisch updatet naar Angular v.20. In de rest van dit document wordt naar deze applicatie verwezen als de “updater”. De updater doorloopt de broncode van een applicatie en maakt een oplijsting van alle nodige aanpassingen en tracht de aanpassing zelf uit te voeren indien mogelijk. De effectiviteit van de updater wordt bepaald aan het aantal gedetecteerde en opgeloste aanpassingen tegenover het totaal van de nodige aanpassingen.

In de volgende sectie wordt een kort overzicht gegeven van de huidige stand van zaken binnen het probleem- en oplossingsdomein. Hierna volgt een beschrijving van de methodologie, waar de werking en evaluatie van de updater in meer detail beschreven is. En tenslotte worden de verwachte resultaten besproken, waarin een inschatting wordt gegeven van de bevindingen van het onderzoek.

2. Literatuurstudie

2.1. Uit te voeren veranderingen

De Angular update guide van Google (2025) geeft een uitgebreid overzicht van alle aanpassingen die nodig zijn voor een Angular applicatie van v.16 naar v.20 te updaten. Uit deze bron blijkt dat in totaal 79 verschillende stappen uitgevoerd worden.

De studie door Di Penta e.a. (2020) onderzoekt welke veranderingen aan code de meeste kans hebben om nieuwe bugs te introduceren. Deze studie maakt het mogelijk om een geïnformeerde inschatting te maken van welke aanpassingen geautomatiseerd kunnen worden zonder ongewenste bijwerkingen te introduceren.

2.2. Automatisatie process

Eén van de meest bekende manieren om code in bulk aan te passen is het gebruikmaken van zoeke en vervangfuncties gebaseerd op reguliere expressies (Regex). Een reguliere expressie is een sequentie van karakters die een patroon in een tekst omschrijft. Aangezien Regex gebaseerd is op tekst, kan het geen rekening houden met de semantiek van de programmeertaal. Uit de studie van Michael e.a. (2019) blijken nog enkele problemen bij de implementatie van Regex, namelijk dat het moeilijk leesbaar, vindbaar, valideerbaar en documenteerbaar is.

Om met de semantiek van de programmeertaal rekening te houden, kan gebruikgemaakt worden van een abstract syntax tree. Zoals omschreven door Sun e.a. (2023), een abstract syntax tree is een datastructuur die de broncode van een applicatie illustreert en rekening houdt met de syntax en semantiek van de programmeertaal. Dit laat ons toe om een stuk code aan te passen, enkel als het in een bepaalde scope zit.

Herinner dat Angular gebaseerd is op de TypeScript programmeertaal. Een bestaande tool voor TypeScript die gebruikmaakt van een abstract syntax tree is de TypeScript Compiler API. De studie door Reid e.a. (2023) onderzoekt hoe de TypeScript compiler gebruikt kan worden voor het corrigeren van foutieve codefragmenten. Deze studie raadt aan om de TypeScript compiler te gebruiken voor statische code-analyse vanwege de effectiviteit, accuraatheid en mogelijkheid om foutieve code te detecteren.

Een alternatief voor de TypeScript compiler dat ook gebruikmaakt van een abstract syntax tree is het TypeScript Language Server Protocol (LSP). Het LSP, als omschreven door Bork en Langer (2023), is een open protocol voor gebruik in verschillende code-editors of integrated development environments (IDEs) dat programmeertaal specifieke functies voorziet zoals: automatisch code aanvullen en code diagnostiek. Dezelfde bron omschrijft LSPs als het de facto standaardprotocol voor de implementatie van deze functies in IDE’s.

Tenslotte is het mogelijk om artificiële intelligentie in te zetten om deze aanpassingen te maken. Met de recente opkomst van AI-tools die specifiek gemaakt zijn voor programmeren, is het mogelijk om deze taak uit te besteden aan AI. Uit de studie door Hodovychenko en Kurinko (2025) blijkt dat AI gedreven tools een gebrek hebben aan transparantie en risico lopen de semantiek van de programmeertaal in de loop van de tijd fout te interpreteren. Verder maakt deze studie de bewering dat voor het maken van dit soort AI-tools er nood is aan een grote hoeveelheid betrouwbare trainingsdata. Dit is data waarover het bedrijf Stater niet beschikt.

3. Methodologie

3.1. Literatuurstudie

Dit onderzoek start met een uitgebreide literatuurstudie naar de verschillen tussen Angular v.16 en Angular v.20. De nodige veranderingen worden opgelijst en onderverdeeld in verschillende categorieën. Deze oplijsting bepaalt de capaci-

teiten van de updater. Verder geeft dit een beter overzicht van welke aanpassingen al dan niet geschikt zijn voor automatisatie.

Verder wordt onderzocht wat de verschillende manieren zijn om automatisch code te updaten. Eén of meerdere manieren worden verkozen om te implementeren op basis van de volgende parameters:

- De complexiteit van de implementatie. Een voorkeur wordt gegeven aan het gebruikmaken van bestaande tools boven het ontwikkelen van nieuwe algoritmes.
- De betrouwbaarheid van de output. Is het mogelijk om met een correcte configuratie een incorrecte output te krijgen?

Deze literatuurstudie neemt één tot twee weken in beslag en het resultaat dient als basis voor de volgende fase van het onderzoek.

3.2. Ontwikkeling van de updater

In deze fase van het onderzoek zal de updater ontwikkeld worden op basis van de voorafgaande literatuurstudie.

De updater heeft drie functies: het detecteren van aanpassingen in de broncode, het evalueren of deze aanpassingen automatisch uitgevoerd kunnen worden, en de aanpassingen uitvoeren indien mogelijk. Elke gedetecteerde aanpassing wordt geregistreerd, waaronder de locatie in de broncode, de aard van de wijziging, de categorie, en of de updater in staat is om de wijziging automatisch uit te voeren.

De updater wordt in een gecontroleerde omgeving getest om de stabiliteit te verzekeren. Voor het ontwikkelen van de applicatie wordt vier tot vijf weken voorzien.

3.3. Evaluatie

De updater wordt uitgevoerd op één van de applicaties binnen Stater. De effectiviteit van de updater wordt bepaald aan het aantal gedetecteerde en opgeloste aanpassingen tegenover het totaal van de nodige aanpassingen.

4. Verwacht resultaat, conclusie

Op basis van de literatuurstudie en de gehanteerde methodologie verwacht dit onderzoek dat minstens 65% van alle nodige aanpassingen automatisch uitgevoerd kan worden. Het automatisatieproces zal naar verwachting de nodige tijd voor de applicaties te updaten verminderen. De hoeveelheid tijd die in totaal bespaard wordt, zal afhangen van het aantal applicaties en de grootte van de broncode die geüpdatet moet worden. Het ontwikkelen van de updater heeft uiteraard ook tijd in beslag genomen. Deze oplossing zal wellicht meer tijd in beslag nemen als enkel één kleine applicatie geüpdatet moet worden.

Of dit de meest effectieve manier is om deze casus op te lossen, is open voor debat. Verder onderzoek zal uitgevoerd moeten worden naar de performantie, accuraatheid en complexiteit van de verschillende implementaties besproken in de literatuurstudie.

Referenties

Bork, D., & Langer, P. (2023). Language server protocol: An introduction to the protocol, its use, and adoption for web modeling tools. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, *18*, 9–1. https://doi.org/10.18417/emisa.18.9

Callaghan, M. D. (2023). Upgrading Angular. In *Angular for Business: Awaken the Advocate Within and Become the Angular Expert at Work* (pp. 95–118). Springer. https://doi.org/10.1007/978-1-4842-9609-7_8

Cincovic, J., Delcev, S., & Draskovic, D. (2019). Architecture of web applications based on Angular Framework: A Case Study. *methodology*, *7*(7), 254–259. https://www.eventiotic.com/eventiotic/files/Papers/URL/df6b5054-816e-4bee-b983-663fb87be2cd.pdf

Di Penta, M., Bavota, G., & Zampetti, F. (2020). On the relationship between refactoring actions and bugs: a differentiated replication. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 556–567. https://doi.org/10.1145/3368089.3409695

Google. (2025). *Angular update guide*. Verkregen september 8, 2025, van https://angular.dev/update-guide?v=16.0-20.0

Hodovychenko, M. A. H. M. A., & Kurinko, D. D. K. D. D. (2025). Analysis of existing approaches to automated refactoring of object-oriented software systems. *Вісник сучасних інформаційних технологій*, *8*(2), 179–196. https://doi.org/10.15276/hait.08.2025.11

Kaur, U., & Singh, G. (2015). A review on software maintenance issues and how to reduce maintenance efforts. *International Journal of Computer Applications*, *118*(1), 6–11. https://doi.org/10.5120/20707-3021

Michael, L. G., Donohue, J., Davis, J. C., Lee, D., & Servant, F. (2019). Regexes are hard: Decision-making, difficulties, and risks in programming regular expressions. *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 415–426. https://doi.org/10.1109/ASE.2019.00047

Reid, B., Treude, C., & Wagner, M. (2023). Using the TypeScript compiler to fix erroneous Node.

js snippets. *2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 220–230. https://doi.org/10.1109/SCAM59687.2023.00031

Sun, W., Fang, C., Miao, Y., You, Y., Yuan, M., Chen, Y., Zhang, Q., Guo, A., Chen, X., Liu, Y., e.a. (2023). Abstract syntax tree for programming language understanding and representation: How far are we? *arXiv preprint arXiv:2312.00413*. https://doi.org/10.48550/arXiv.2312.00413

Vaniea, K., & Rashidi, Y. (2016). Tales of software updates: The process of updating software. *Proceedings of the 2016 chi conference on human factors in computing systems*, 3215–3226. https://doi.org/10.1145/2858036.2858303