Bachelorproef toegepaste informatica (2025-2026)

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

Wauters Sander
sander.wauters@student.hogent.be

Promotor: Irina Malfait
Co-promotor: Peter De Seranno (Stater)

Hogeschool Gent, Arbeidstraat 14, 9300 Aalst
Keuzerichting: Mobile & Enterprise development

Sleutelwoorden: Angular, TypeScript, Static code analysis, Automatisatie

Samenvatting

Applicatie ontwikkeld om de update van Angular-applicaties te ondersteunen. Getest om een Angular v16 applicatie te updaten naar v20. De evaluatie toont aan dat ongeveer 25% van alle nodige de aanpassingen automatisch kon worden uitgevoerd. Deze aanpak is bruikbaar voor het refactoren van verschillende TypeScript-applicaties. Programmatische configuratie zorgt voor een hoge flexibiliteit. De oplossing is gebaseerd op de TypeScript-compiler en gebruikt dezelfde foutdetectie.

Introductie

Het Angular-framework vereenvoudigt het ontwikkelingsproces voor het bouwen van dynamische webapplicaties. Zoals bij de meeste software ontvangt Angular regelmatig updates. Deze updates zijn noodzakelijk, omdat ze de cyberveiligheid verbeteren. Het toepassen van dergelijke updates is echter niet altijd vanzelfsprekend.

Angular verwijdert verouderde functionaliteiten uit het framework. Daardoor moet de broncode van Angular-applicaties aangepast worden. Dit type updates vindt om de 6 maanden plaats. Bij meerdere enterprise-applicaties kan de benodigde tijd voor dit updateproces snel oplopen. Het bedrijf Stater ervaart dit probleem. Stater is een end-to-end dienstverlener voor zowel hypothecaire als consumentenkredieten. Zij willen al hun applicaties updaten van Angular v16 naar v20. De sprong van vier versies betekent dat er vermoedelijk veel aanpassingen in de broncode nodig zijn.

Methodologie

Deze studie begon met een literatuurstudie naar gekende methodes om code automatisch aan te passen. De beslissing werd gemaakt om een *updater* te maken op basis van de TypeScript Compiler API. Door de TypeScript-compiler programmatisch aan te spreken en te kopellen aan zoek- en vervangfuncties was hetmogelijk om gericht code op te sporen en te veranderen.

Een collectie aan helperfuncties zijn ontwikkeld voor de implementatie te vereenvoudigen. De updater wordt programmatische ingesteld. Dit geeft ons een hoge flexibiliteit waaroor deze werkwijze bij verschillende soorten aanpassingen bruikbaar is. De updater zelf is een simpel commandline applicatie. Het kan op deze manier op meerdere projecten opgeroepen worden. Het codeframent toont hoe we alle instanties van een functie gericht van naam kunnen veranderen doorheen meerdere applicaties.

```ts
const project = loadProject();
project.getSourceFiles().forEach((file) ⇒
  findNodes(
    file,
    (node) ⇒
      deepestInstanceOf(node, "async") &&
      node.getKind() ≠= SyntaxKind.AsyncKeyword,
    (node) ⇒ node.replaceWithText("waitForAsync"),
  ),
);
await saveProject(project);
```

Onderzoeksvragen

In welke mate kan de automatisering van het updateproces van Angular v16 naar v20, bij meerdere applicaties, de onderhoudstijd voor de ontwikkelaars verlagen?
Deelvragen:

- Welke veranderingen moeten uitgevoerd worden om Angular van v16 naar v20 te updaten?
- Welke manieren bestaan er 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?
- Wat zijn statistisch gezien de meest voorkomende problemen bij het updaten van code?

Conclusies

De testresultaten tonen aan dat de updater een meerwaarde biedt in de ondersteuning van het updateproces. Ondanks dat het verwachte resultaat niet bereikt is, was het nog steeds mogelijk om een vierde van alle aanpassingen automatisch uit te voeren. De hoge flexibiliteit van deze aanpak maakt het mogelijk voor de updater te herconfigureren voor toekomstige updates. Bovendien is de updater niet gelimiteerd aan het uitvoeren van Angular-updates. Dezelfde manier van werken kan toegepast worden om meer algemene refactoringen uit te voeren. Verder kan men de updater configureren om op andere TypeScript-applicaties te werken.

Categorie	Totaal		TypeScript		Testen		Syntax		Semantiek	
#Stappen	80	100,00%	50	100,00%	16	100,00%	24	100,00%	38	100,00%
Automatiseerbaar										
Volledig	22	27,50%	10	20,00%	3	18,75%	9	37,50%	1	2,63%
Deels	8	10,00%	8	16,00%	0	0,00%	5	20,83%	5	13,16%
Niet	50	62,50%	32	64,00%	13	81,25%	10	41,67%	32	84,21%
Detecteerbaar										
Volledig	24	30,00%	24	48,00%	6	37,50%	12	50,00%	13	34,21%
Deels	5	6,25%	5	10,00%	0	0,00%	2	8,33%	4	10,53%
Niet	51	63,75%	21	42,00%	10	62,50%	10	41,67%	21	55,26%

Resultaten van de updater om een Angular-applicatie te updaten van v16 naar v20.

Toekomstig onderzoek

In de stand van zaken hebben we verschillende automatisatietechnieken besproken. Een vergelijking van deze technieken voor toepassing in andere casussen kan best interessant zijn. Wanneer zou een integratie met een language server, of met AI, gepast zijn bijvoorbeeld?

Tijdens het schrijven van dit onderzoek is Angular v21 uitgekomen. Deze versie komt met nieuwe tools om AI beter te integreren in het ontwikkelingsproces. Voornamelijk beweren ze dat het AI toelaat om de nieuwste functionaliteiten te gebruiken. Dit was één van de redenen dat AI niet gekozen werd in dit onderzoek. Deze tools zijn momenteel nog experimenteel, maar kunnen veelbelovend zijn.

Broncode: `https://github.com/Sander-Wauters/hogent-2526-thesis-SanderWauters`