

Titel van de bachelorproef.

Optionele ondertitel.

Sarah Bader.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. L. Blondeel

Co-promotor: Dhr. B. Tjassens Keiser

Academiejaar: 2025–2026

Eerste examenperiode

Departement IT en Digitale Innovatie .

**HO
GENT**

Woord vooraf

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

Lijst van figuren	vii
Lijst van tabellen	viii
Lijst van codefragmenten	ix
1 Inleiding	1
1.1 De onderzoeksvraag	2
1.2 Het doel	2
1.3 Het verwachte resultaat.	3
1.4 Structuur van het onderzoek	3
2 Stand van zaken	4
2.01 Belang van de mainframecomputer	4
2.02 Nood aan onderhoud van HLASM-code	4
2.03 Moeilijkheden voor beginnende ontwikkelaars	5
2.04 Effectiviteit van visualisatietechnieken.	5
2.05 Bestaande visualisatietools en parsers.	5
2.06 Control Flow in High Level Assembler	5
2.07 JavaScript Parsing	6
2.08 Mermaid.js en flowcharts	6
3 Methodologie	12
3.01 Fase 1: Oriëntatie en Literatuurstudie	12
3.02 Fase 2: Requirementsanalyse.	13
3.03 Fase 3: Implementatie en Evaluatie.	13
3.04 Fase 4: Resultaat en Verdediging	14
3.05 Tijdsplanning	14
4 Conclusie	17
A Onderzoeksvoorstel	19
A.1 Inleiding	19
A.2 Literatuurstudie	20
A.2.1 Belang van de mainframecomputer	20
A.2.2 Nood aan onderhoud van HLASM-code	21
A.2.3 Moeilijkheden voor beginnende ontwikkelaars	21
A.2.4 Effectiviteit van visualisatietechnieken.	21

A.2.5	Bestaande visualisatietools en parsers	21
A.2.6	Control Flow in High Level Assembler	22
A.2.7	JavaScript Parsing	22
A.2.8	Mermaid.js en flowcharts	23
A.3	Methodologie	23
A.3.1	Fase 1: Oriëntatie en Literatuurstudie	23
A.3.2	Fase 2: Requirementsanalyse.	24
A.3.3	Fase 3: Implementatie en Evaluatie.	24
A.3.4	Fase 4: Resultaat en Verdediging	25
A.3.5	Tijdsplanning	25
A.4	Verwacht resultaat, conclusie	25

Bibliografie	27
---------------------	-----------

Lijst van figuren

2.1	Voorbeeld figuur.	7
3.1	Gantt-diagram met een tijdsindicatie per onderzoeksfase.	15
A.1	Gantt-diagram met een tijdsindicatie per onderzoeksfase.	26

Lijst van tabellen

2.1 Voorbeeld tabel	11
-------------------------------	----

Lijst van codefragmenten

2.1 Voorbeeld codefragment	8
----------------------------------	---

1

Inleiding

1.1. Probleemstelling

Vaak wordt een mainframe gezien als verouderde technologie die enkel gekend is uit films. Het tegengestelde is waar: mainframes zijn één van de meest geavanceerde servers die voortdurend onderhouden en gemoderniseerd worden. Mainframes zijn dan ook nog volop in gebruik. Door de betrouwbaarheid en capaciteit van een mainframe is het vooral geschikt voor grote aantallen transacties/aanvragen van gebruikers. Mainframes worden vaak gebruikt door banken, verzekeraars, luchtvaartmaatschappijen... om hun fundamentele bedrijfsprocessen te ondersteunen. Omdat deze systemen zo fundamenteel zijn, is het onderhoud ervan van groot belang. Vaak zijn de programma's die draaien op het mainframe ook tientallen jaren oud. Een deel van deze programma's is geprogrammeerd in High Level Assembler (HLASM). Deze programmeertaal wordt vaak gekozen omdat het programma snel moet zijn en directe controle over de hardware nodig heeft. Andere programma's maken vaak gebruik van de HLASM-programma's, waardoor ze ervan afhankelijk worden.

Daarbij zijn deze programma's vaak minimaal gedocumenteerd of bevatten ze verouderde documentatie. Hierdoor is het moeilijk om als nieuwe mainframe developer de voorgaande code te begrijpen.

Veel bedrijven hadden het idee dat mainframes onnodig zouden worden en dat ze konden vertrouwen op "modernere" servers. In werkelijkheid groeide hun workload alleen maar en werd die te groot om op reguliere servers te runnen. Met de voortdurende modernisatie bleek het mainframe echter de beste keuze om hun bedrijfsactiviteiten verder op te vertrouwen.

Met de oude mindset dat ze van het mainframe zouden weggaan, investeerden bedrijven niet in nieuw mainframe talent. Hierdoor bestaat de industrie vooral uit professionals die binnenkort met pensioen gaan. Nu het duidelijk is dat een main-

frame beter in staat is om hun huidige workload te beheren, is het tekort aan junior talent een probleem. De kennis van de bedrijfsprogramma's en systemen zit dus in een generatie die de arbeidsmarkt verlaat.

Hierdoor is er nood aan een nieuwe generatie om vooral het onderhoud van de systemen en programma's te beheren. Door het gebrek aan voldoende en bruikbare documentatie is het vaak moeilijk om als nieuwe mainframe programmeur code te begrijpen, wat het onderhouden ervan belemmert. Documentatie is dus belangrijk voor de leesbaarheid en onderhoudbaarheid van de code, waardoor minder fouten worden gemaakt.

Een hulpmiddel zou een visualisatietool kunnen zijn die kan dienen als documentatie. Beginners hebben vaak moeite met het begrijpen van de control flow van een programma, daarom is visualisatie van de control flow vooral handig.

1.2. De onderzoeksvraag

Dit onderzoek is gericht op het beantwoorden van de volgende vraag:

Hoe effectief versnellen statische analyse en Mermaid-diagrammen het leerproces van HLASM bij beginners?

Binnen dit onderzoek wordt gefocust op twee aspecten: het technische en het menselijke. Er wordt gekeken naar hoe de visualisatie technisch kan geïmplementeerd worden en ook naar de effectiviteit van de visualisatie als documentatie en hulpmiddel bij het begrijpen van de control flow in HLASM.

1.3. Onderzoeksdoelstelling

Het doel van dit onderzoek is het uitwerken van een Proof-of-Concept en een bijhorende analyse ervan. De PoC moet in staat zijn de belangrijkste onderdelen van HLASM-code om te zetten in een Mermaid.js flowchart. De focus ligt expliciet op de structuur van het programma, dus minder op de details. Het moet bijvoorbeeld weergeven waar het programma begint, waar het zich opsplijt in verschillende takken en waar het eindigt.

Vooraleer de implementatie van start gaat, wordt onderzoek gedaan naar de bestaande technologieën die van hulp kunnen zijn. Dit onderzoek wordt uitgevoerd in de vorm van een literatuurstudie. Tijdens de literatuurstudie wordt gekeken naar het probleemdomen en het oplossingsdomen.

Binnen de literatuurstudie worden volgende vragen gesteld om het probleemdomen beter te verstaan:

- Waarom is een mainframe nog steeds van belang?
- Waarom is er nood aan het onderhouden van HLASM-code?
- Hoe helpt visualisatie met het begrijpen van codestructuur?

- Wat zijn de moeilijkheden bij het begrijpen van assemblytalen voor beginnende ontwikkelaars?

Rond het oplossingsdomein worden volgende vragen bestudeerd:

- Hoe helpt visualisatie bij het sneller begrijpen van programma's?
- Welke tools bestaan er al om HLASM-code te visualiseren?
- Welke HLASM parsers bestaan er?
- Wat is de beste manier om de kolom-gebaseerde structuur van HLASM te verwerken met een JavaScript-parser?
- Welke JavaScript-libraries zijn toegankelijk en bruikbaar?

1.4. Het verwachte resultaat

Het verwachte resultaat van dit onderzoek is dat de visualisatie helpt bij het beheersen van HLASM-control flow, in tegenstelling tot het gebruik van manuele methodes. Om dit resultaat te meten wordt de PoC geëvalueerd aan de hand van testen die de tijd meten met de PoC en met de manuele manier.

1.5. Structuur van het onderzoek

Deze bachelorproef is als volgt opgesteld:

In Hoofdstuk 2 (Literatuurstudie) worden de deelvragen uit het oplossingsdomein en het probleemdomein onderzocht aan de hand van bestaande onderzoeken en literatuur. Hiermee wordt een beter zicht gecreëerd op het probleem en de oplossing om zo gericht mogelijk te werken.

In Hoofdstuk 3 (Methodologie) wordt toegelicht welke onderzoekstechnieken worden gebruikt.

In Hoofdstuk 4 (HLASM-analyse) wordt gekeken naar Assembler, hoe de programmeertaal is opgebouwd en welke elementen de structuur van het programma vastleggen.

In Hoofdstuk 5 (Control-flow visualisatie) wordt bekeken hoe de Assembler-structuur kan worden omgezet naar Mermaid.js-elementen. Er wordt onderzocht hoe dit visueel logisch kan worden gerepresenteerd.

In Hoofdstuk 6 (Proof-of-Concept) wordt de PoC uitgewerkt en daarna geëvalueerd aan de hand van testen.

In Hoofdstuk 7 (Conclusie) wordt gekeken naar het resultaat van de evaluatie en naar de literatuurstudie om een antwoord te formuleren op de onderzoeksvragen.

2

Stand van zaken

2.0.1. Belang van de mainframecomputer

Vaak worden mainframes beschreven als oude technologie en wordt gedacht dat ze niet meer relevant zijn. Ook wouden bedrijven graag mainframes vervangen door andere servers. Momenteel zijn mainframes nog steeds zeer actief en staan ze in voor kritische bedrijfsactiviteiten. Tegenwoordig worden mainframes ook geïntegreerd met moderne technologieën (Sagers e.a., [2018](#)). Het onderzoek van Sagers e.a. ([2018](#)) wordt bevestigd door recent onderzoek waarin wordt aangetoond dat organisaties niet enkel gebruikmaken van legacyapplicaties, maar ook bezig zijn met het actief ontwikkelen van nieuwe applicaties (Shaik, [2024](#)).

Mainframes zijn dus een zeer relevante en moderne technologie, die dagelijks draait. Omdat ervaren mainframeprofessionals hun pensioen naderen is er een behoefte aan een nieuwe generatie mainframespecialisten die het onderhoud van deze systemen kunnen waarborgen (Waites & Ketterer, [2013](#)).

2.0.2. Nood aan onderhoud van HLASM-code

Bedrijfskritische functionaliteiten van mainframeapplicaties zijn vaak afhankelijk van High Level Assembler (HLASM), waardoor onderhoud van deze legacycode van groot belang is. Als de aangeroepen HLASM-code faalt, dan faalt het volledige programma (Zaytsev, [2020](#)). Uit een onderzoek naar softwareonderhoud blijkt dat het begrijpen van bestaande code een groot deel van de moeite neemt tijdens onderhoud. Vooral bij beginnende ontwikkelaars gaat het merendeels van hun tijd aan het begrijpen van de functionaliteit, dit is omdat zij minder ervaring hebben (Xia e.a., [2018](#)). Dit is vooral het geval bij legacycode zoals HLASM, omdat daar de documentatie vaak beperkt is.

2.0.3. Moeilijkheden voor beginnende ontwikkelaars

Assemblytalen zijn zeer gedetailleerd, als ontwikkelaar definieer je elke kleine stap van het programma, wat ingewikkeld kan zijn voor beginners. Uit onderzoek blijkt dat naarmate de ervaring toeneemt, het gemakkelijker wordt om de functionaliteit van code te begrijpen (Feigenspan e.a., 2012).

Binnen het onderwijs hebben studenten vooral moeite met de componenten die de logische volgorde van het programma bepalen. Dit zijn de componenten die de control flow van het programma definiëren. Bij het begrijpen van de control flow maken beginners vaak fouten en ervaren ze vaak verwarring (Kawash, 2024). Hieruit kan worden geconcludeerd dat de control flow van code een moeilijkheid is onder beginnende ontwikkelaars.

2.0.4. Effectiviteit van visualisatietechnieken

Visualisatietechnieken worden vaak gebruikt om code beter te begrijpen en worden ook erkend als technieken die sneller en meer inzicht geven in codefunctionaliteiten. Visualisaties worden vaak gebruikt om de structuur van code te representeren (Storey, 2005). Bij assemblytalen kan visualisatie vooral helpen, aangezien de code vaak is opgebouwd uit zeer gedetailleerde instructies en kan worden gegroepeerd in componenten (Baldwin e.a., 2009).

2.0.5. Bestaande visualisatietools en parsers

Een bestaande tool die kan worden gebruikt voor de visualisatie van HLASM-code is FermaT. Deze tool is vooral gericht op het begrijpen en migreren van legacycode. Het is een zeer krachtige en zware tool die veel functionaliteiten heeft (Ward, 2001). Ook biedt IBM een HLASM parser aan, die onderdeel maakt van IBM Developer for z/OS. Deze parser is bedoeld voor diepe analyse en gebruik met andere tools (IBM, g.d.).

Het doel van dit onderzoek is het ontwikkelen van een lichte tool die HLASM-code omzet naar een high-level control-flowdiagram visualisatie in Mermaid.js-syntax. Deze tool is specifiek bedoeld voor educatieve doeleinden, om beginnende ontwikkelaars te ondersteunen. Ook ligt de focus enkel op een beperkte subset van control-flow-instructies, in tegenstelling tot bestaande tools.

2.0.6. Control Flow in High Level Assembler

High Level Assembler (HLASM) wordt gezien als een Second Generation Language (2GL), de instructies in HLASM komen rechtstreeks overeen met processor-commando's. Er zijn dus geen abstracties zoals functies of objecten. Deze abstracties zijn wel terug te vinden in talen met een hoger generatieniveau zoals COBOL. HLASM biedt daardoor veel controle over het fysieke systeem, maar dat maakt de code vaak moeilijker te lezen en te onderhouden (Kornelis, 2003).

Om de control-flow van HLASM te definiëren kan er gefocust worden op bepaalde

elementen die sprongen of vertakkingen maken:

- Een directe sprong: Instructies zoals B (Branch) sturen het programma direct naar een ander codeblok (*High Level Assembler for z/OS & z/VM & z/VSE: HLASM VIR6 Programmer's Guide*, [g.d.](#)).
- Een conditionele sprong: Door middel van vergelijkingen (C voor Compare) of condities (BE voor Branch on Equal) verandert de route van het programma (*High Level Assembler for z/OS & z/VM & z/VSE: HLASM VIR6 Programmer's Guide*, [g.d.](#)).
- Een subroutine: Bij het aanroepen van een subroutine worden instructies zoals BAL of BAS gebruikt. Het terugkeren vanuit een subroutine gebeurt via een BR-instructie (Ward, [2013](#)).

Deze specifieke instructies kunnen een idee geven van de control-flow van het programma.

2.0.7. JavaScript Parsing

Binnen HLASM wordt er gewerkt met een vaste kolom-gebaseerde indeling. Elk onderdeel van de code staat op een vaste plek. Elke kolom heeft een eigen functie: kolommen 1-8 zijn voor labels, kolommen 10-14 voor de instructie en vanaf kolom 16 staan de operanden. Na kolom 71 wordt alles beschouwd als commentaar of wordt het genegeerd (*HLASM VIR6 Language Reference*, [g.d.](#)).

JavaScript is zeer nuttig om tekst te parsen, het is namelijk erg goed in het doorzoeken en herkennen van patronen (Flanagan, [2011](#)). Voor de PoC is een regel-voor-regel aanpak voldoende. Hierbij zal de parser elke regel doorlopen op basis van de vaste kolomposities. Zo kunnen de labels en instructies herkend worden die bijdragen aan de control-flow. De verzamelde elementen worden vervolgens gemapt naar Mermaid.js-elementen.

2.0.8. Mermaid.js en flowcharts

Met Mermaid.js worden flowcharts en andere structuren vanuit tekst automatisch gevisualiseerd (Sveidqvist & Jain, [2021](#)). Dit maakt het makkelijk om een mapping te maken van HLASM-elementen naar een visuele representatie. Het gebruik van flowcharts om complexe software begrijpelijk te maken is een bekend middel in de informatica (Ensmenger, [2016](#)).

Een flowchart maakt gebruik van blokken en pijlen om een route te definiëren. In de context van informatica worden flowcharts vaak gebruikt om de route van een programma weer te geven (Charntaweekhun & Wangsiripitak, [2006](#)). In het geval van HLASM kunnen bijvoorbeeld spronginstructies worden weergegeven als pijlen die het programma brengen naar een ander codeblok.



Figuur 2.1: Voorbeeld van invoegen van een figuur. Zorg altijd voor een uitgebreid bijschrift dat de figuur volledig beschrijft zonder in de tekst te moeten gaan zoeken. Vergeet ook je bronvermelding niet!

Dit hoofdstuk bevat je literatuurstudie. De inhoud gaat verder op de inleiding, maar zal het onderwerp van de bachelorproef **diepgaand** uitspitten. De bedoeling is dat de lezer na lezing van dit hoofdstuk helemaal op de hoogte is van de huidige stand van zaken (state-of-the-art) in het onderzoeksdomein. Iemand die niet vertrouwd is met het onderwerp, weet nu voldoende om de rest van het verhaal te kunnen volgen, zonder dat die er nog andere informatie moet over opzoeken (Pollefliet, 2011).

Je verwijst bij elke bewering die je doet, vakterm die je introduceert, enz. naar je bronnen. In \LaTeX kan dat met het commando `\textcite{}` of `\autocite{}`. Als argument van het commando geef je de “sleutel” van een “record” in een bibliografische databank in het Bib \LaTeX -formaat (een tekstbestand). Als je expliciet naar de auteur verwijst in de zin (narratieve referentie), gebruik je `\textcite{}`. Soms is de auteursnaam niet expliciet een onderdeel van de zin, dan gebruik je `\autocite{}` (referentie tussen haakjes). Dit gebruik je bv. bij een citaat, of om in het bijschrift van een overgenomen afbeelding, broncode, tabel, enz. te verwijzen naar de bron. In de volgende paragraaf een voorbeeld van elk.

Knuth (1998) schreef een van de standaardwerken over sorteer- en zoekalgoritmen. Experts zijn het erover eens dat cloud computing een interessante opportuniteit vormen, zowel voor gebruikers als voor dienstverleners op vlak van informatietechnologie (Creager, 2009).

Let er ook op: het `cite`-commando voor de punt, dus binnen de zin. Je verwijst meteen naar een bron in de eerste zin die erop gebaseerd is, dus niet pas op het einde van een paragraaf.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In


```
1 import pandas as pd
2 import seaborn as sns
3
4 penguins = sns.load_dataset('penguins')
5 sns.relplot(data=penguins, x="flipper_length_mm",
    ↪ y="bill_length_mm", hue="species")
```

Codefragment 2.1: Voorbeeld van het invoegen van een codefragment.

hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetur at, consectetur sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetur odio sem sed wisi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetur eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetur tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at,

dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Kolom 1	Kolom 2	Kolom 3
α	β	γ
A	10.230	a
B	45.678	b
C	99.987	c

Tabel 2.1: Voorbeeld van een tabel.

3

Methodologie

Deze bachelorproef is een toegepast en technisch onderzoek waarin een oplossing wordt gezocht voor het onderhouden en begrijpen van legacy HLASM-broncode. Het onderzoek start met een literatuurstudie naar het probleemdomain en het oplossingsdomain. Ook wordt de control flow van HLASM-broncode bestudeerd, waarna een subset van representatieve HLASM-instructies en labels geselecteerd worden. Na de literatuurstudie start de ontwikkeling van een technische proof-of-concept (PoC) die de geselecteerde subset visueel omzet in een control-flowchart in Mermaid.js-syntax.

3.0.1. Fase 1: Oriëntatie en Literatuurstudie

Tijdens deze fase wordt gefocust op het aanscherpen van de probleemstelling en het oplossingsdomain. Dit gebeurt door middel van een literatuurstudie waarin dieper wordt ingegaan op de deelvragen:

- Wat is het belang van de mainframecomputer?
- Waarom is er nood aan het onderhouden van HLASM-code?
- Wat zijn de moeilijkheden bij het begrijpen van assemblytalen voor beginnende ontwikkelaars?
- Wat is de effectiviteit van visualisatietechnieken voor het sneller begrijpen van programma's?
- Wat zijn de bestaande visualisatietools binnen High Level Assembler?
- Welke HLASM-instructies zijn van noodzaak om de control-flow van de code in kaart te brengen?

- Wat is de beste manier om de kolom-gebaseerde structuur van HLASM te verwerken met een JavaScript-parser?
- Hoe vertaal je de vertakkingen in de HLASM-code naar correcte Mermaid.js-diagrammen?

Deze fase loopt van 20 december 2025 tot 1 maart 2026 en resulteert in een uitgewerkte literatuurstudie en methodologie. Het resultaat bevat concreet een theoretisch onderzoek naar de probleemstelling en oplossingsdomein, en een afgebakende scope van HLASM-instructies en labels voor de verdere ontwikkeling van de technische oplossing.

3.0.2. Fase 2: Requirementsanalyse

Tijdens de literatuurstudie (tussen januari 2026 en maart 2026) wordt een requirementsanalyse uitgevoerd. Het doel van deze fase is om concrete en meetbare eisen voor de proof of concept en de evaluatie vast te leggen.

Binnen deze fase beginnen we eerst met het specifiek definiëren van de doelgroep. Uit de doelgroep wordt gezocht naar 6 à 10 deelnemers die instaan voor de evaluatie van de PoC. Voor de evaluatie worden ook HLASM-codefragmenten verzameld die typische control-flowconstructies hebben zoals labels, conditionele takken en subroutines-aanroepen.

Ook worden de succescriteria van de PoC vastgesteld:

- correcte detectie van de geselecteerde HLASM-instructies en labels;
- correcte omzetting naar Mermaid.js-flowcharts;
- meetbare tijdswinst bij het begrijpen van de control flow in vergelijking met een manuele analyse.

3.0.3. Fase 3: Implementatie en Evaluatie

Tijdens de implementatie- en evaluatiefase wordt de proof-of-concept uitgewerkt en iteratief geëvalueerd. Deze fase vindt plaats tussen 2 maart 2026 en 4 mei 2026. De ontwikkeling gebeurt volgens een scrum-werkwijze om de effectiviteit van de PoC te waarborgen. Met behulp van de subset die in de vorige fase werd gedefiniëerd, wordt een JavaScript-gebaseerde parser ontwikkeld die HLASM-code statisch analyseert. De parser identificeert instructies en labels aan de hand van patroonherkenning in de structuur van HLASM-broncode en zet deze via een mapping om naar Mermaid.js-syntax.

Concreet worden tijdens deze fase de volgende aspecten ontwikkeld:

- de logische opbouw van de parser op basis van patroonherkenning in de HLASM-broncodestructuur;

- een mapping van de geselecteerde HLASM-instructies en labels naar Mermaid.js-elementen;
- een evaluatie van de PoC op correctheid en tijdswinst.

Tijdens de ontwikkeling wordt de PoC iteratief geëvalueerd via een vergelijkende studie met handmatige code reviews. De evaluatie wordt uitgevoerd door 6 à 10 vrijwillige deelnemers, de helft van de deelnemers gebruikt de PoC om codefragmenten uit te leggen en de andere helft doet dit zonder de PoC. Beide groepen maken gebruik van dezelfde codefragmenten en worden getimed op het moment waarop zij zelf aangeven dat ze de code begrijpen.

De evaluatie loopt tijdens de implementatie, van 1 april 2026 tot 4 mei 2026.

3.0.4. Fase 4: Resultaat en Verdediging

De laatste fase loopt van 4 mei 2026 tot juni 2026. In deze fase worden de onderzoeksresultaten en evaluatieresultaten verwerkt tot een conclusie.

Specifiek worden de volgende aspecten besproken:

- de sterktes en beperkingen van de PoC;
- mogelijke uitbreidingen voor toekomstig onderzoek.

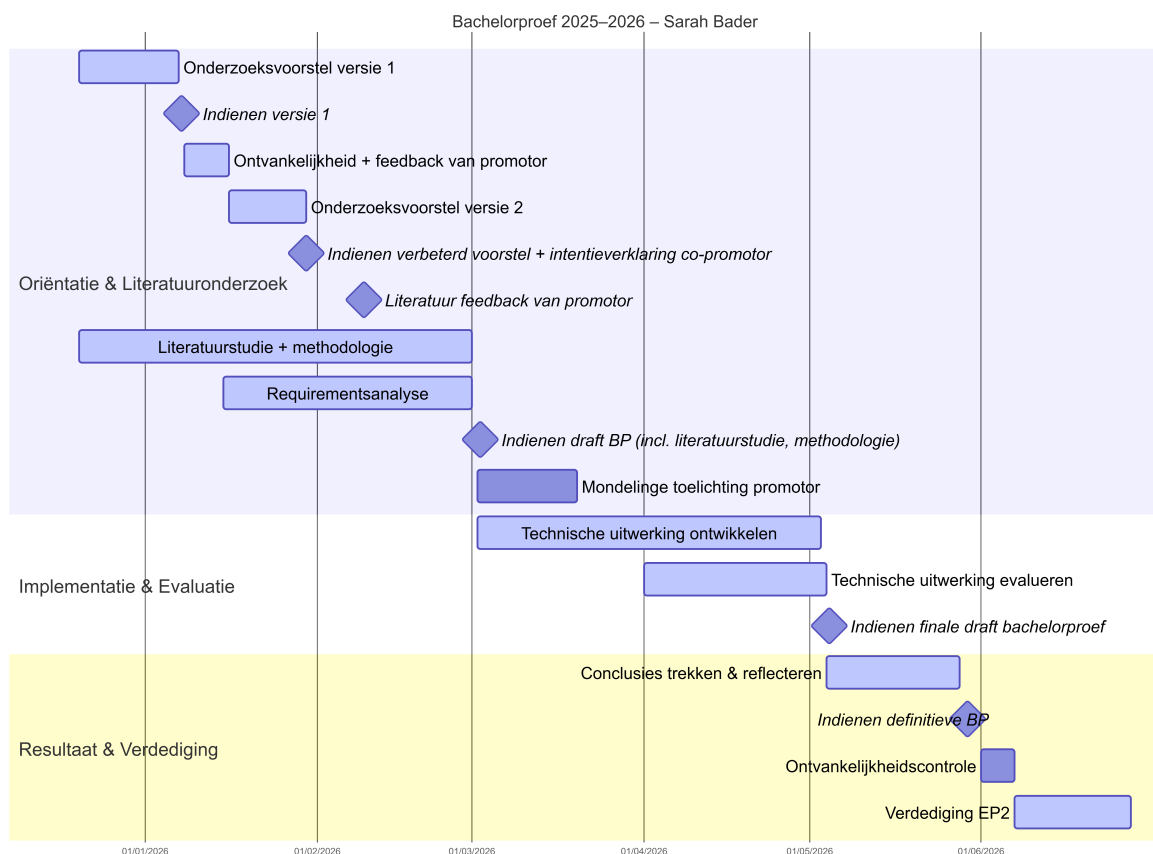
Deze fase resulteert in de finale bachelorproef die wordt ingediend op 29 mei 2026, de verdediging en de proof-of-concept met de bijbehorende documentatie.

3.0.5. Tijdsplanning

Elke fase heeft een tijdsperiode waarin de specifieke resultaten moeten worden opgeleverd voor de volgende fase. Zoals te zien in figuur A.1, zijn ook de administratieve deliverables opgenomen.

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetur quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta.



Figuur 3.1: Gantt-diagram met een tijdsindicatie per onderzoeksfase.

Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

4

Conclusie

Het verwachte resultaat van deze studie is een proof of concept (PoC), die in staat is een subset van representatieve HLASM- en labels om te zetten in visuele stappen geïnterpreteerd in Mermaid.js-syntax. De PoC brengt door middel van statische HLASM-broncodeanalyse de control flow in kaart. Met deze PoC wordt verwacht beginnende mainframeontwikkelaars te ondersteunen met het begrijpen van een onbekend stuk HLASM-broncode. De evaluatie van het verwachte resultaat wordt uitgevoerd door middel van een vergelijking met de traditionele, handmatige manieren van code reviews.

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

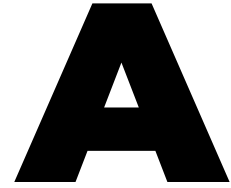
Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor

malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consecutur libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.



Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Inleiding

Tegenwoordig spelen mainframecomputers een cruciale rol in de dagelijkse activiteiten van bedrijven in de financiële sector, de gezondheidszorg, verzekeringen en tal van andere publieke en private ondernemingen. Deze bedrijven zijn afhankelijk van mainframecomputers en hebben dan ook de nood om deze zo optimaal mogelijk te onderhouden. Vermits veel ervaren mainframe-specialisten hun pensioen naderen, is er nood aan een nieuwe generatie mainframe-specialisten die het onderhoud van hun systemen kan waarborgen. Onder dit onderhoud vallen de bedrijfskritische functionaliteiten van mainframeapplicaties, die voornamelijk afhankelijk zijn van High-Level Assembler (HLASM). Deze legacycode is vaak tientallen jaren oud en minimaal gedocumenteerd. Voor de nieuwe generatie mainframeontwikkelaars vormt dit een uitdaging om deze legacycode te begrijpen en te onderhouden. Zij vertrouwen vaak enkel op manuele code reviews.

Door middel van een geautomatiseerde tool die statische analyse van HLASM-code visualiseert in een control-flowchart, wil dit onderzoek ondersteuning bieden aan beginnende mainframeontwikkelaars, specifiek binnen organisaties die afhankelijk zijn van legacy HLASM-code.

De probleemstelling van dit onderzoek luidt als volgt:

Hoe effectief versnellen statische analyse en Mermaid-diagrammen het leerproces van HLASM bij beginners?

Het doel van dit onderzoek is het ontwikkelen van een proof-of-concept (PoC) die

een representatieve subset van HLASM-instructies en labels statisch analyseert en deze omzet in een visueel control-flowdiagram in Mermaid.js-syntax.

Vooraleer de ontwikkeling van de PoC van start gaat, wordt een literatuurstudie uitgevoerd naar het probleemdomein en het oplossingsdomein.

Rond het probleemdomein worden volgende vragen bestudeerd:

- Wat is het belang van de mainframecomputer?
- Waarom is er nood aan het onderhouden van HLASM-code?
- Wat zijn de moeilijkheden bij het begrijpen van assemblytalen voor beginnende ontwikkelaars?

Rond het oplossings domein worden volgende vragen bestudeerd:

- Hoe helpt visualisatie bij het sneller begrijpen van programma's?
- Welke tools bestaan er al om HLASM-code te visualiseren?
- Welke HLASM parsers bestaan er?
- Welke HLASM-instructies zijn van noodzaak om de control-flow van de code in kaart te brengen?
- Wat is de beste manier om de kolom-gebaseerde structuur van HLASM te verwerken met een JavaScript-parser?
- Hoe vertaal je de vertakkingen in de HLASM-code naar correcte Mermaid.js-diagrammen?

A.2. Literatuurstudie

A.2.1. Belang van de mainframecomputer

Vaak worden mainframes beschreven als oude technologie en wordt gedacht dat ze niet meer relevant zijn. Ook wouden bedrijven graag mainframes vervangen door andere servers. Momenteel zijn mainframes nog steeds zeer actief en staan ze in voor kritische bedrijfsactiviteiten. Tegenwoordig worden mainframes ook geïntegreerd met moderne technologieën (Sagers e.a., 2018). Het onderzoek van Sagers e.a. (2018) wordt bevestigd door recent onderzoek waarin wordt aangetoond dat organisaties niet enkel gebruikmaken van legacyapplicaties, maar ook bezig zijn met het actief ontwikkelen van nieuwe applicaties (Shaik, 2024).

Mainframes zijn dus een zeer relevante en moderne technologie, die dagelijks draait. Omdat ervaren mainframeprofessionals hun pensioen naderen is er een behoefte aan een nieuwe generatie mainframespecialisten die het onderhoud van deze systemen kunnen waarborgen (Waites & Ketterer, 2013).

A.2.2. Nood aan onderhoud van HLASM-code

Bedrijfskritische functionaliteiten van mainframeapplicaties zijn vaak afhankelijk van High Level Assembler (HLASM), waardoor onderhoud van deze legacycode van groot belang is. Als de aangeroepen HLASM-code faalt, dan faalt het volledige programma (Zaytsev, 2020). Uit een onderzoek naar softwareonderhoud blijkt dat het begrijpen van bestaande code een groot deel van de moeite neemt tijdens onderhoud. Vooral bij beginnende ontwikkelaars gaat het merendeels van hun tijd aan het begrijpen van de functionaliteit, dit is omdat zij minder ervaring hebben (Xia e.a., 2018). Dit is vooral het geval bij legacycode zoals HLASM, omdat daar de documentatie vaak beperkt is.

A.2.3. Moeilijkheden voor beginnende ontwikkelaars

Assemblytalen zijn zeer gedetailleerd, als ontwikkelaar definieer je elke kleine stap van het programma, wat ingewikkeld kan zijn voor beginners. Uit onderzoek blijkt dat naarmate de ervaring toeneemt, het gemakkelijker wordt om de functionaliteit van code te begrijpen (Feigenspan e.a., 2012).

Binnen het onderwijs hebben studenten vooral moeite met de componenten die de logische volgorde van het programma bepalen. Dit zijn de componenten die de control flow van het programma definiëren. Bij het begrijpen van de control flow maken beginners vaak fouten en ervaren ze vaak verwarring (Kawash, 2024). Hieruit kan worden geconcludeerd dat de control flow van code een moeilijkheid is onder beginnende ontwikkelaars.

A.2.4. Effectiviteit van visualisatietechnieken

Visualisatietechnieken worden vaak gebruikt om code beter te begrijpen en worden ook erkend als technieken die sneller en meer inzicht geven in codefunctionaliteiten. Visualisaties worden vaak gebruikt om de structuur van code te representeren (Storey, 2005). Bij assemblytalen kan visualisatie vooral helpen, aangezien de code vaak is opgebouwd uit zeer gedetailleerde instructies en kan worden gegroepeerd in componenten (Baldwin e.a., 2009).

A.2.5. Bestaande visualisatietools en parsers

Een bestaande tool die kan worden gebruikt voor de visualisatie van HLASM-code is FermaT. Deze tool is vooral gericht op het begrijpen en migreren van legacycode. Het is een zeer krachtige en zware tool die veel functionaliteiten heeft (Ward, 2001). Ook biedt IBM een HLASM parser aan, die onderdeel maakt van IBM Developer for z/OS. Deze parser is bedoeld voor diepe analyse en gebruik met andere tools (IBM, g.d.).

Het doel van dit onderzoek is het ontwikkelen van een lichte tool die HLASM-code omzet naar een high-level control-flowdiagram visualisatie in Mermaid.js-syntax. Deze tool is specifiek bedoeld voor educatieve doeleinden, om beginnende ont-

wikkelaars te ondersteunen. Ook ligt de focus enkel op een beperkte subset van control-flow-instructies, in tegenstelling tot bestaande tools.

A.2.6. Control Flow in High Level Assembler

High Level Assembler (HLASM) wordt gezien als een Second Generation Language (2GL), de instructies in HLASM komen rechtstreeks overeen met processor-commando's. Er zijn dus geen abstracties zoals functies of objecten. Deze abstracties zijn wel terug te vinden in talen met een hoger generatieniveau zoals COBOL. HLASM biedt daardoor veel controle over het fysieke systeem, maar dat maakt de code vaak moeilijker te lezen en te onderhouden (Kornelis, 2003).

Om de control-flow van HLASM te definiëren kan er gefocust worden op bepaalde elementen die sprongen of vertakkingen maken:

- Een directe sprong: Instructies zoals B (Branch) sturen het programma direct naar een ander codeblok (*High Level Assembler for z/OS & z/VM & z/VSE: HLASM V1R6 Programmer's Guide*, [g.d.](#)).
- Een conditionele sprong: Door middel van vergelijkingen (C voor Compare) of condities (BE voor Branch on Equal) verandert de route van het programma (*High Level Assembler for z/OS & z/VM & z/VSE: HLASM V1R6 Programmer's Guide*, [g.d.](#)).
- Een subroutine: Bij het aanroepen van een subroutine worden instructies zoals BAL of BAS gebruikt. Het terugkeren vanuit een subroutine gebeurt via een BR-instructie (Ward, 2013).

Deze specifieke instructies kunnen een idee geven van de control-flow van het programma.

A.2.7. JavaScript Parsing

Binnen HLASM wordt er gewerkt met een vaste kolom-gebaseerde indeling. Elk onderdeel van de code staat op een vaste plek. Elke kolom heeft een eigen functie: kolommen 1-8 zijn voor labels, kolommen 10-14 voor de instructie en vanaf kolom 16 staan de operanden. Na kolom 71 wordt alles beschouwd als commentaar of wordt het genegeerd (*HLASM V1R6 Language Reference*, [g.d.](#)).

JavaScript is zeer nuttig om tekst te parsen, het is namelijk erg goed in het doorzoeken en herkennen van patronen (Flanagan, 2011). Voor de PoC is een regel-voor-regel aanpak voldoende. Hierbij zal de parser elke regel doorlopen op basis van de vaste kolomposities. Zo kunnen de labels en instructies herkend worden die bijdragen aan de control-flow. De verzamelde elementen worden vervolgens gemapt naar Mermaid.js-elementen.

A.2.8. Mermaid.js en flowcharts

Met Mermaid.js worden flowcharts en andere structuren vanuit tekst automatisch gevisualiseerd (Sveidqvist & Jain, 2021). Dit maakt het makkelijk om een mapping te maken van HLASM-elementen naar een visuele representatie. Het gebruik van flowcharts om complexe software begrijpelijk te maken is een bekend middel in de informatica (Ensmenger, 2016).

Een flowchart maakt gebruik van blokken en pijlen om een route te definiëren. In de context van informatica worden flowcharts vaak gebruikt om de route van een programma weer te geven (Charntaweechun & Wangsiripitak, 2006). In het geval van HLASM kunnen bijvoorbeeld spronginstructies worden weergegeven als pijlen die het programma brengen naar een ander codeblok.

A.3. Methodologie

Deze bachelorproef is een toegepast en technisch onderzoek waarin een oplossing wordt gezocht voor het onderhouden en begrijpen van legacy HLASM-broncode. Het onderzoek start met een literatuurstudie naar het probleemdomein en het oplossingsdomein. Ook wordt de control flow van HLASM-broncode bestudeerd, waarna een subset van representatieve HLASM-instructies en labels geselecteerd worden. Na de literatuurstudie start de ontwikkeling van een technische proof-of-concept (PoC) die de geselecteerde subset visueel omzet in een control-flowchart in Mermaid.js-syntax.

A.3.1. Fase 1: Oriëntatie en Literatuurstudie

Tijdens deze fase wordt gefocust op het aanscherpen van de probleemstelling en het oplossingsdomein. Dit gebeurt door middel van een literatuurstudie waarin dieper wordt ingegaan op de deelvragen:

- Wat is het belang van de mainframecomputer?
- Waarom is er nood aan het onderhouden van HLASM-code?
- Wat zijn de moeilijkheden bij het begrijpen van assemblytalen voor beginnende ontwikkelaars?
- Wat is de effectiviteit van visualisatietechnieken voor het sneller begrijpen van programma's?
- Wat zijn de bestaande visualisatietools binnen High Level Assembler?
- Welke HLASM-instructies zijn van noodzaak om de control-flow van de code in kaart te brengen?
- Wat is de beste manier om de kolom-gebaseerde structuur van HLASM te verwerken met een JavaScript-parser?

- Hoe vertaal je de vertakkingen in de HLASM-code naar correcte Mermaid.js-diagrammen?

Deze fase loopt van 20 december 2025 tot 1 maart 2026 en resulteert in een uitgewerkte literatuurstudie en methodologie. Het resultaat bevat concreet een theoretisch onderzoek naar de probleemstelling en oplossingsdomein, en een afgebakende scope van HLASM-instructies en labels voor de verdere ontwikkeling van de technische oplossing.

A.3.2. Fase 2: Requirementsanalyse

Tijdens de literatuurstudie (tussen januari 2026 en maart 2026) wordt een requirementsanalyse uitgevoerd. Het doel van deze fase is om concrete en meetbare eisen voor de proof of concept en de evaluatie vast te leggen.

Binnen deze fase beginnen we eerst met het specifiek definiëren van de doelgroep. Uit de doelgroep wordt gezocht naar 6 à 10 deelnemers die instaan voor de evaluatie van de PoC. Voor de evaluatie worden ook HLASM-codefragmenten verzameld die typische control-flowconstructies hebben zoals labels, conditionele takken en subroutines-aanroepen.

Ook worden de succescriteria van de PoC vastgesteld:

- correcte detectie van de geselecteerde HLASM-instructies en labels;
- correcte omzetting naar Mermaid.js-flowcharts;
- meetbare tijdswinst bij het begrijpen van de control flow in vergelijking met een manuele analyse.

A.3.3. Fase 3: Implementatie en Evaluatie

Tijdens de implementatie- en evaluatiefase wordt de proof-of-concept uitgewerkt en iteratief geëvalueerd. Deze fase vindt plaats tussen 2 maart 2026 en 4 mei 2026. De ontwikkeling gebeurt volgens een scrum-werkwijze om de effectiviteit van de PoC te waarborgen. Met behulp van de subset die in de vorige fase werd gedefinieerd, wordt een JavaScript-gebaseerde parser ontwikkeld die HLASM-code statisch analyseert. De parser identificeert instructies en labels aan de hand van patroonherkenning in de structuur van HLASM-broncode en zet deze via een mapping om naar Mermaid.js-syntax.

Concreet worden tijdens deze fase de volgende aspecten ontwikkeld:

- de logische opbouw van de parser op basis van patroonherkenning in de HLASM-broncodestructuur;
- een mapping van de geselecteerde HLASM-instructies en labels naar Mermaid.js-elementen;

- een evaluatie van de PoC op correctheid en tijdswinst.

Tijdens de ontwikkeling wordt de PoC iteratief geëvalueerd via een vergelijkende studie met handmatige code reviews. De evaluatie wordt uitgevoerd door 6 à 10 vrijwillige deelnemers, de helft van de deelnemers gebruikt de PoC om codefragmenten uit te leggen en de andere helft doet dit zonder de PoC. Beide groepen maken gebruik van dezelfde codefragmenten en worden getimed op het moment waarop zij zelf aangeven dat ze de code begrijpen.

De evaluatie loopt tijdens de implementatie, van 1 april 2026 tot 4 mei 2026.

A.3.4. Fase 4: Resultaat en Verdediging

De laatste fase loopt van 4 mei 2026 tot juni 2026. In deze fase worden de onderzoeksresultaten en evaluatieresultaten verwerkt tot een conclusie.

Specifiek worden de volgende aspecten besproken:

- de sterktes en beperkingen van de PoC;
- mogelijke uitbreidingen voor toekomstig onderzoek.

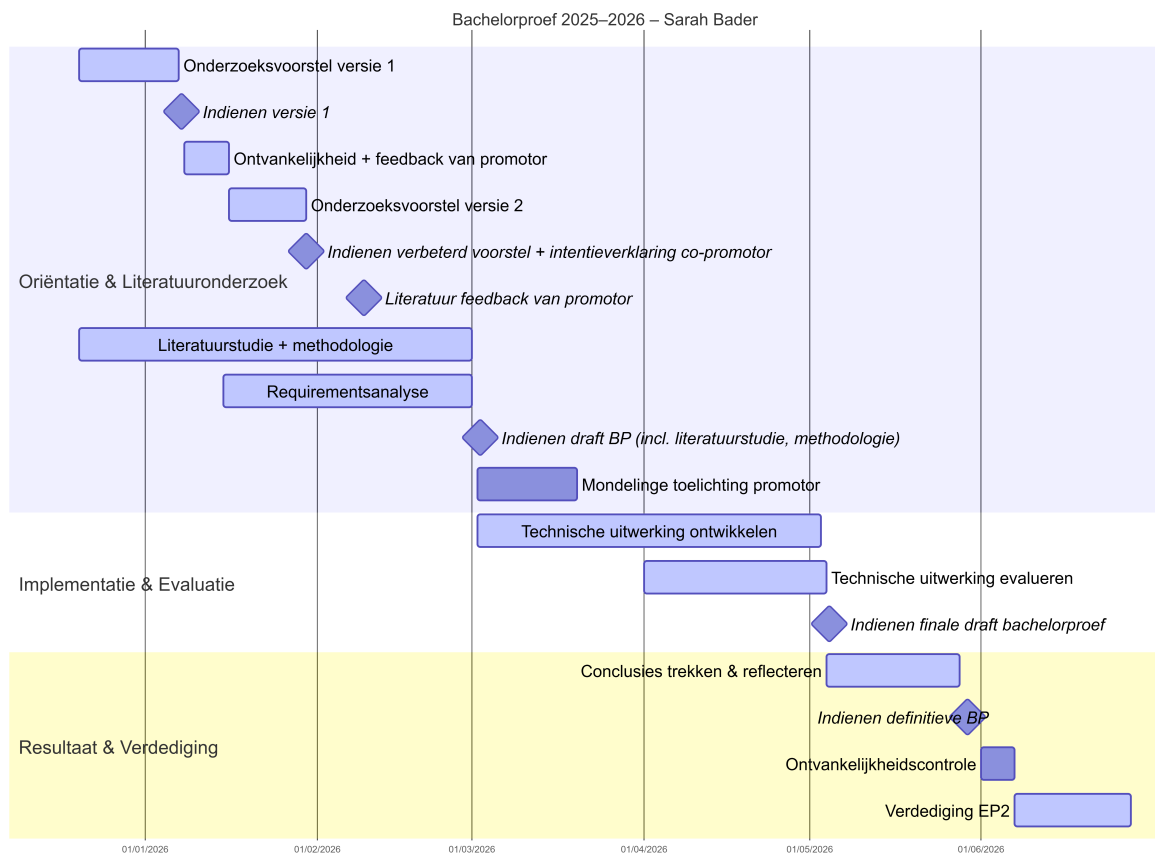
Deze fase resulteert in de finale bachelorproef die wordt ingediend op 29 mei 2026, de verdediging en de proof-of-concept met de bijbehorende documentatie.

A.3.5. Tijdsplanning

Elke fase heeft een tijdsperiode waarin de specifieke resultaten moeten worden opgeleverd voor de volgende fase. Zoals te zien in figuur [A.1](#), zijn ook de administratieve deliverables opgenomen.

A.4. Verwacht resultaat, conclusie

Het verwachte resultaat van deze studie is een proof of concept (PoC), die in staat is een subset van representatieve HLASM- en labels om te zetten in visuele stappen geïnterpreteerd in Mermaid.js-syntax. De PoC brengt door middel van statische HLASM-broncodeanalyse de control flow in kaart. Met deze PoC wordt verwacht beginnende mainframeontwikkelaars te ondersteunen met het begrijpen van een onbekend stuk HLASM-broncode. De evaluatie van het verwachte resultaat wordt uitgevoerd door middel van een vergelijking met de traditionele, handmatige manieren van code reviews.



Figuur A.1: Gantt-diagram met een tijdsindicatie per onderzoeksfase.

Bibliografie

- Baldwin, J., Myers, D., Storey, M.-A., & Coady, Y. (2009). Assembly Visualization and Analysis: An Old Dog CAN Learn New Tricks! <https://ecs.wgtn.ac.nz/foswiki/pub/Events/PLATEAU/2009Program/plateau09-baldwin.pdf>
- Charntaweechun, K., & Wangsiripitak, S. (2006). Visual Programming using Flowchart. *2006 International Symposium on Communications and Information Technologies*, 1062–1065. <https://doi.org/10.1109/ISCIT.2006.339940>
- Creeger, M. (2009). CTO Roundtable: Cloud Computing. *Communications of the ACM*, 52(8), 50–56.
- Ensmenger, N. (2016). The Multiple Meanings of a Flowchart. *51*(3), 321–351. Verkregen januari 25, 2026, van <http://www.jstor.org/stable/44667617>
- Feigenspan, J., Kästner, C., Liebig, J., Apel, S., & Hanenberg, S. (2012). Measuring programming experience. *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, 73–82. <https://doi.org/10.1109/ICPC.2012.6240511>
- Flanagan, D. (2011). *JavaScript: The Definitive Guide* (6th). O'Reilly Media.
- High Level Assembler for z/OS & z/VM & z/VSE: HLASM VIR6 Programmer's Guide* [SC26-4941-07]. (g.d.). IBM Corporation.
- HLASM VIR6 Language Reference* [SC26-4940-06]. (g.d.). IBM Corporation.
- IBM. (g.d.). *HLASM Parser* [IBM Developer for z/OS documentation]. Verkregen januari 27, 2026, van <https://www.ibm.com/docs/en/developer-for-zos/16.0.x?topic=files-hlasm-parser>
- Kawash, J. (2024). Where Do Students Struggle Most in a First Course on Assembly Language? *Proceedings of the 26th Western Canadian Conference on Computing Education*. <https://doi.org/10.1145/3660650.3660652>
- Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc.
- Kornelis, A. F. (2003). HLASM - Why assembler? <https://bixoft.nl/english/why.htm>
- Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Academia Press.
- Sagers, G., Ball, K., Hosack, B., Twitchell, D., & Wallace, D. (2018). The Mainframe Is Dead. Long Live the Mainframe! *AIS Transactions on Enterprise Systems*, 2(1). <https://www.aes-journal.com/index.php/ais-tes/article/view/6>
- Shaik, S. (2024). Importance of Mainframe in Modern Era of Technologies. *Journal of Engineering and Applied Sciences Technology*, 1–3. [https://doi.org/https://doi.org/10.47363/jeast/2024\(6\)257](https://doi.org/https://doi.org/10.47363/jeast/2024(6)257)

- Storey, M.-A. (2005). Theories, methods and tools in program comprehension. *IEEE Software*, 22(3), 36–45.
- Sveidqvist, K., & Jain, A. (2021). *The official guide to Mermaid.js: Create complex diagrams and beautiful flowcharts easily using text and code*. Packt Publishing.
- Waites, S., & Ketterer, J. (2013). Lack of Mainframe Programmers a Critical Issue for IT Organizations A Review of the Literature. *International Journal of Business, Humanities and Technology*, 3(7). https://ijbht.thebrpi.org/journals/Vol_3_No_7_September_2013/3.pdf
- Ward, M. (2013). Assembler restructuring in FermaT. *2013 IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 147–156. <https://doi.org/10.1109/SCAM.2013.6648196>
- Ward, M. (2001). The FermaT assembler re-engineering workbench. *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, 659–662. <https://doi.org/10.1109/ICSM.2001.972783>
- Xia, X., Bao, L., Lo, D., Xing, Z., Hassan, A. E., & Li, S. (2018). Measuring Program Comprehension: A Large-Scale Field Study with Professionals. *IEEE Transactions on Software Engineering*, 44(10), 951–976. <https://doi.org/10.1109/TSE.2017.2734091>
- Zaytsev, V. (2020). Journal of Object Technology Published by AITO -Association Internationale pour les Technologies Objets Vadim Zaytsev. Modelling of Language Syntax and Semantics: The Case of the Assembler Compiler. *Journal of Object Technology*, 19(1), 1–22. <https://grammarware.net/text/2020/hlasm.pdf>