

De beste masterproef ooit

Eerste Auteur
Tweede Auteur

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
elektrotechniek, optie Elektronica en
chipontwerp

Promotor

Prof. dr. ir. Weet Beter

Evaluatoren

Ir. W. Eetveel

W. Eetrest

Begeleiders

Ir. Een Assistent

E. Vriend

© 2026 KU Leuven – Faculteit Ingenieurswetenschappen
Uitgegeven in eigen beheer door Eerste Auteur en Tweede Auteur,
Faculteit Ingenieurswetenschappen, Kasteelpark Arenberg 1 bus 2200, B-3001 Leuven

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever. Deze uitgave bevat het studiewerk van een student in het kader van de academische opleiding en examenbeoordeling. Na deze beoordeling vond geen correctie plaats van het studiewerk.

Voorwoord

Ik wil iedereen bedanken die mij het afgelopen jaar bezig heeft gehouden, in het bijzonder mijn promotor en mijn begeleiders. Ik wil ook de jury bedanken voor het lezen van de tekst. Mijn oprechte dank gaat ook uit naar mijn partner en de rest van mijn familie.

Eerste Auteur
Tweede Auteur

Inhoudsopgave

Voorwoord	i
Samenvatting	iii
Lijst van figuren en tabellen	iv
Lijst van afkortingen en symbolen	v
Inleiding	1
1 Gezamenlijke bewerking van documenten	3
1.1 Eisen aan het systeem	4
2 Benaderingen voor synchronisatie	5
2.1 Operationele transformatie	5
2.2 Conflictvrije gerepliceerde datatypen	6
3 Ontwerp en implementatie	7
3.1 Opslag van de geschiedenis	7
4 Evaluatie	9
4.1 Bespreking van de resultaten	10
Conclusie	11
Literatuurlijst	13
Transparantieverklaring GenAI-gebruik	15

Samenvatting

De samenvatting geeft een uitgebreider overzicht van het werk, maar blijft beperkt tot één pagina.

Lijst van figuren en tabellen

Lijst van figuren

1.1	Typisch verloop van het gezamenlijk schrijven van een document	4
4.1	Aantal bewerkingen per dag van de week en uur in de onderzochte sessies	10

Lijst van tabellen

4.1	Gemiddelde latentie en omvang van de opgeslagen gegevens naar aantal gebruikers	9
-----	---	---

Lijst van afkortingen en symbolen

Afkortingen

LoG	Laplacian-of-Gaussian
MSE	Gemiddelde kwadratische fout
PSNR	Piek-signaal-ruisverhouding

Symbolen

c	Lichtsnelheid
E	Energie
m	Massa
π	Het getal pi

Inleiding

Tekstdocumenten worden tegenwoordig nog maar zelden door één enkele auteur geschreven. Studenten schrijven werkstukken in tweetallen, ontwikkelaars onderhouden samen technische documentatie en onderzoeksteams leggen op het laatste moment vanaf meerdere locaties tegelijk de laatste hand aan artikelen. Om dergelijk schrijven mogelijk te maken, moet de editor waarborgen dat alle deelnemers dezelfde tekst zien en dat geen enkele wijziging verloren gaat.

Het doel van dit werk is twee hoofdbenaderingen voor de synchronisatie van documenten bij gezamenlijke bewerking in realtime te vergelijken, operationele transformatie en conflictvrije gerepliceerde datatypen, en deze te toetsen aan de hand van een prototype van een webeditor. Het eerste hoofdstuk bakt de eisen aan een dergelijk systeem af, het tweede beschrijft beide benaderingen, het derde het ontwerp van het prototype en het vierde de meetresultaten.

Hoofdstuk 1

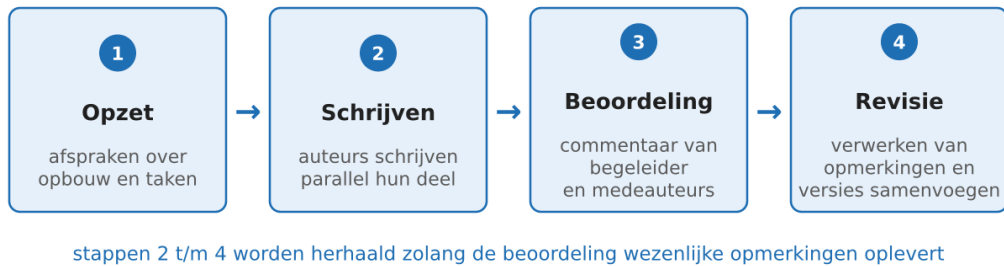
Gezamenlijke bewerking van documenten

Onder gezamenlijke bewerking verstaan we de situatie waarin meerdere gebruikers hetzelfde document bewerken en hun wijzigingen bij elkaar doorwerken zonder handmatig samenvoegen. Naargelang de snelheid waarmee wijzigingen zich verspreiden, onderscheiden we drie basale werkwijzen:

- **asynchrone bewerking**, waarbij auteurs volledige versies van het document aan elkaar doorgeven en deze pas achteraf samenvoegen,
- **bewerking met vergrendeling**, waarbij op een gegeven moment slechts één auteur een bepaald deel van het document mag bewerken,
- **bewerking in realtime**, waarbij elke wijziging onmiddellijk zichtbaar wordt voor de andere gebruikers.

Dit werk richt zich uitsluitend op de laatstgenoemde werkwijze. Deze is voor de gebruiker het prettigst, maar stelt de hoogste eisen aan het systeemontwerp, omdat wijzigingen gelijktijdig op veel plaatsen ontstaan en in verschillende volgorde binnenkomen.

Bewerking in realtime verandert daarbij niet alleen de techniek, maar ook het verloop van de hele samenwerking. Het typische verloop van gezamenlijk schrijven, zoals de geïnterviewde gebruikers het beschreven, is weergegeven in Figuur 1.1.



FIGUUR 1.1: Typisch verloop van het gezamenlijk schrijven van een document

1.1 Eisen aan het systeem

Uit interviews met gebruikers van bestaande editors en uit de vakliteratuur hebben we vier eisen afgeleid waaraan een systeem voor gezamenlijke bewerking moet voldoen:

1. convergentie, dat wil zeggen dat na aflevering van alle wijzigingen alle kopieën van het document dezelfde inhoud hebben,
2. behoud van intentie, dat wil zeggen dat het resultaat van een bewerking overeenkomt met wat de auteur op het moment van bewerken zag,
3. responsiviteit, dat wil zeggen dat een lokale bewerking onmiddellijk zichtbaar is zonder op de server te wachten,
4. robuustheid, dat wil zeggen dat een kortstondige verbindingsonderbreking niet leidt tot verlies van tekst waaraan wordt gewerkt.

De eerste twee eisen betreffen correctheid, de overige twee bruikbaarheid. In de praktijk blijkt juist de combinatie van onmiddellijke respons en gegarandeerde convergentie het moeilijkste deel van het hele ontwerp.

Hoofdstuk 2

Benaderingen voor synchronisatie

Systemen voor gezamenlijke bewerking in realtime voeren elke wijziging eerst lokaal uit en verspreiden deze pas daarna onder de andere deelnemers. Ze verschillen vooral in de manier waarop ze omgaan met de situatie waarin twee wijzigingen gelijktijdig ontstaan en elke deelnemer ze in een andere volgorde ontvangt.

2.1 Operationele transformatie

Operationele transformatie gaat terug op het werk van Ellis en Gibbs [1], die als eersten beschreven hoe een gelijktijdige operatie zo kan worden aangepast dat zij ook kan worden uitgevoerd na een andere operatie waarvan haar auteur niet op de hoogte was. Elke bewerking wordt vastgelegd als een invoeg- of verwijderoperatie op een bepaalde positie in de tekst.

Als bijvoorbeeld de ene gebruiker een woord aan het begin van een alinea invoegt en de andere tegelijkertijd een woord aan het einde ervan verwijdert, moet de positie van de verwijdering worden verschoven met de lengte van het ingevoegde woord. Deze verschuiving wordt berekend door de transformatiefunctie.

2.1.1 Transformatiefunctie

De transformatiefunctie krijgt een paar gelijktijdige operaties en geeft daarvan aangepaste versies terug. Om alle kopieën van het document te laten convergeren, moet de functie voldoen aan de zogeheten transformatie-eigenschappen, waarvan de geldigheid voor algemene operaties moeilijk te bewijzen is. De meeste systemen in de praktijk vertrouwen daarom op een centrale server die één enkele volgorde van operaties vastlegt.

2.2 Conflictvrije gerepliceerde datatypen

De tweede benadering wordt gevormd door conflictvrije gerepliceerde datatypen, gezamenlijk aangeduid met de afkorting CRDT [2]. In plaats van operaties achteraf aan te passen, kiezen zij een zodanige representatie van het document dat gelijktijdige operaties commuteren en hun volgorde geen invloed heeft op het resultaat.

Een gerepliceerd datatype is conflictvrij als twee willekeurige replica's die dezelfde verzameling updates hebben ontvangen dezelfde toestand hebben, ongeacht de volgorde waarin die zijn afgeleverd.

Elk teken van het document krijgt een unieke identifier die gedurende het hele bestaan van het teken gelijk blijft. De invoegpositie wordt dan niet bepaald door een getal, maar door een verwijzing naar de identifier van het naburige teken, zodat gelijktijdige bewerkingen elders in de tekst haar niet kunnen verschuiven.

Hoofdstuk 3

Ontwerp en implementatie

Het prototype van de editor bestaat uit een webclient, een synchronisatieserver en een database waarin de server doorlopend de toestand van de documenten opslaat. Zowel de client als de server is zo geschreven dat de synchronisatielaag kan worden vervangen en beide beschreven benaderingen op dezelfde gebruikersinterface kunnen worden vergeleken.

De client houdt een lokale replica van het document bij en zet elke bewerking van de gebruiker om in een operatie op het gerepliceerde datatype. De operaties worden in batches naar de server gestuurd, zodat het netwerk bij snel typen niet overbelast raakt. De vereenvoudigde verwerking van een binnenkomende batch op de server is weergegeven in het volgende fragment:

```
def verwerk_batch(document, batch):
    for operatie in batch.operaties:
        if operatie.id in document.toegepast:
            continue
        document.pas_toe(operatie)
        document.toegepast.add(operatie.id)
    stuur_naar_anderen(document.id, batch)
```

Omdat de operaties commuteren, hoeft de server de batches niet te ordenen of te transformeren. Het volstaat dat hij ervoor zorgt dat geen enkele operatie twee keer wordt uitgevoerd en dat elke batch alle verbonden clients bereikt.

3.1 Opslag van de geschiedenis

De server slaat niet alleen de uiteindelijke tekst op, maar het volledige operatielogboek. Uit het logboek kan elke eerdere versie van het document worden hersteld en kan de toestand van de replica van een client die langere tijd offline was opnieuw worden opgebouwd. Deze aanpak komt overeen met een architectuur op basis van een onveranderlijk gebeurtenissenlogboek, zoals beschreven door Kleppmann [3]. Om

3. ONTWERP EN IMPLEMENTATIE

te voorkomen dat het logboek onbeperkt groeit, comprimeert de server het met regelmatige tussenpozen tot snapshots.

Hoofdstuk 4

Evaluatie

We hebben het prototype vergeleken met een referentie-implementatie van operationele transformatie op een verzameling registraties van echte bewerkingssessies. Voor elke sessie maten we de latentie, dat wil zeggen de tijd tussen een bewerking bij de auteur en de weergave ervan bij de andere deelnemers, en de gemiddelde latentie van de sessie berekenden we volgens (4.1):

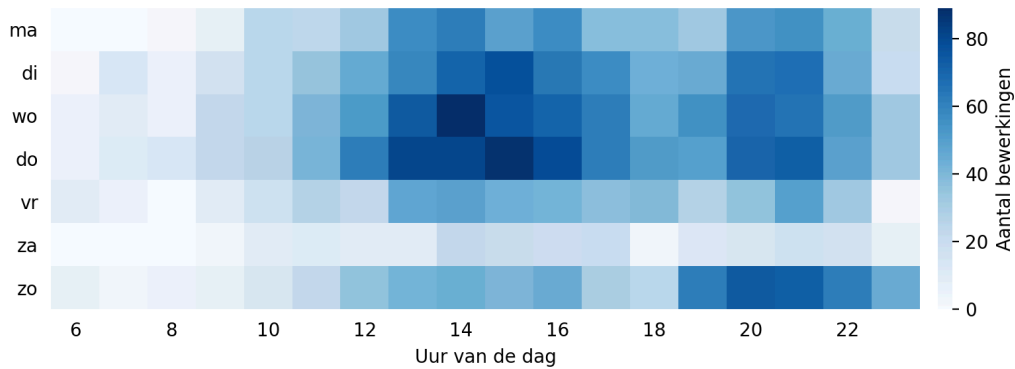
$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{aflevering}} - t_i^{\text{bewerking}}) \quad (4.1)$$

waarbij n het aantal operaties in de sessie is. De gemeten waarden voor verschillende aantallen gelijktijdig werkende gebruikers zijn samengevat in Tabel 4.1.

TABEL 4.1: Gemiddelde latentie en omvang van de opgeslagen gegevens naar aantal gebruikers

Gebruikers	Latentie OT (ms)	Latentie CRDT (ms)	Data (kB)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

Naarmate het aantal gebruikers toeneemt, stijgt de latentie van operationele transformatie aanzienlijk sneller, omdat de server elke operatie moet transformeren ten opzichte van alle gelijktijdige operaties. De prijs voor de lagere latentie van CRDT is een grotere hoeveelheid opgeslagen gegevens, aangezien elk teken zijn eigen identificator meedraagt. De verdeling van de bewerkingen in de tijd, waarop de belastingtest is gebaseerd, is weergegeven in Figuur 4.1.



FIGUUR 4.1: Aantal bewerkingen per dag van de week en uur in de onderzochte sessies

4.1 Bespreking van de resultaten

De resultaten bevestigen dat beide benaderingen voor kleinere teams gelijkwaardig zijn en dat veeleer de eenvoud van de implementatie de doorslag geeft. Bij een groter aantal gelijktijdige gebruikers heeft CRDT een duidelijk voordeel, zowel in latentie als in robuustheid tegen verbindingsonderbrekingen. Aanvullende interviews met gebruikers, afgenomen volgens de principes van kwalitatief onderzoek [4], lieten zien dat gebruikers een verschil in latentie van minder dan honderd milliseconden niet waarnemen, maar het verlies van tekst waaraan werd gewerkt na een verbindingsonderbreking als een ernstig probleem beschouwen.

Conclusie

Dit werk heeft operationele transformatie en conflictvrije gerepliceerde datatypen vergeleken als twee benaderingen voor de synchronisatie van documenten bij gezamenlijke bewerking in realtime. Met een prototype van een webeditor hebben we aangetoond dat CRDT's een eenvoudigere server mogelijk maken zonder centrale ordening van operaties en bij grotere aantallen gebruikers een lagere latentie bieden.

Het belangrijkste nadeel van CRDT's blijft de grotere hoeveelheid opgeslagen gegevens. Een volgende stap is daarom na te gaan hoe de omvang van de gegevens de periodieke compressie van het logboek tot snapshots beïnvloedt en hoe het prototype zich gedraagt bij langdurig gebruik met tienduizenden documenten.

Literatuurlijst

- [1] C. A. Ellis en S. J. Gibbs, “Concurrency control in groupware systems”, in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York: ACM, 1989, pp. 399-407. doi: [10.1145/67544.66963](https://doi.org/10.1145/67544.66963).
- [2] M. Shapiro, N. Preguiça, C. Baquero, en M. Zawirski, “Conflict-free replicated data types”, in *Stabilization, Safety, and Security of Distributed Systems*, Berlin: Springer, 2011, pp. 386-400. doi: [10.1007/978-3-642-24550-3_29](https://doi.org/10.1007/978-3-642-24550-3_29).
- [3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol: O’Reilly Media, 2017.
- [4] J. Hendl, *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.

Transparantieverklaring voor het gebruik van GenAI door KU Leuven-studenten (academiejaar 2026–2027)

- Vul de transparantieverklaring in aan de hand van de Gedragscode voor GenAI-gebruik door studenten en de principes en richtlijnen (kuleuven.be/genai) voor GenAI-gebruik aan KU Leuven.
- **Specifieke instructies van je docent of faculteit hebben voorrang** op de universiteitsbrede gedragscode en richtlijnen.
- Door het invullen van dit document erken je de gedragscode en richtlijnen te hebben gelezen en te hebben gebruikt voor het invullen van deze transparantieverklaring.
- Je vinkt hieronder voor elk type gebruik ja/nee aan. In de onderste rij van de tabel vul je het GenAI-gebruik in dat niet onder de overige categorieën valt (of er een combinatie van is).
- Waar de gedragscode stelt dat je niet hoeft te verwijzen naar een bepaald type GenAI-gebruik, heeft dit betrekking op verwijzingen in je werk zelf. Het betrokken GenAI-gebruik vermeld je uiteraard wel in deze transparantieverklaring.

Naam student: _____

Studentennummer: _____

Geef aan met “X” of het om een opdracht of bachelor-/masterproef gaat:

☐ Deze verklaring hoort bij een **opdracht**.

Naam opleidingsonderdeel: _____

Vakcode: _____

Deze verklaring hoort bij mijn ☐ bachelorproef of ☐ masterproef.

Titel proef: _____

Promotor: _____

Dagelijkse begeleider: _____

Geef aan met “X”:

☐ Ik heb geen GenAI-tool gebruikt.☐ Ik heb een GenAI-tool gebruikt. In dit geval, geef aan welke tool(s) (bijv. Copilot, ChatGPT, ...) en omschrijf waar nodig je gebruik:

GenAI-tool gebruikt als/voor:	Ja/Neen	Naam van de gebruikte GenAI-tool. Indien nodig, omschrijf hoe je GenAI gebruikte; gebruik hiervoor de gedragscode en richtlijnen.
Als een zoekmachine om informatie te vinden over een topic of om hierover bestaand onderzoek te vinden	☐ Ja ☐ Neen	
Voor het genereren van nieuwe (onderzoeks)ideeën	☐ Ja ☐ Neen	
Voor literatuuronderzoek (<i>opzoeken bronnen</i>)	☐ Ja ☐ Neen	
Als taalhulp voor het nakijken of verbeteren van zelfgeschreven teksten	☐ Ja ☐ Neen	

GenAI-tool gebruikt als/voor:	Ja/Neen	Naam van de gebruikte GenAI-tool.
Als vertaalhulp om teksten te verbeteren die je zelf schreef of om teksten van anderen beter te begrijpen	☐ Ja ☐ Neen	
Als parafraseertool (herformuleren van woorden uit een zin)	☐ Ja ☐ Neen	
Voor het genereren van programmeercode	☐ Ja ☐ Neen	
Voor het genereren van synthetische data	☐ Ja ☐ Neen	
Voor het genereren van beeldmateriaal, video of audio	☐ Ja ☐ Neen	
Voor het genereren van tekstblokken	☐ Ja ☐ Neen	
Overig gebruik (het kan ook gaan om een combinatie van bovenstaande vormen)	☐ Ja ☐ Neen	