HO
GENT

Faculteit Bedrijf en Organisatie

Gedecentraliseerd en transparant versiebeheer aan de hand van blockchain principes en IPFS: een praktische toepassing voor open source bedrijven.

Michiel Schoofs

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

Promotor:
Karine Samyn
Co-promotor:
Maurice Dalderup

Instelling: —

Academiejaar: 2019-2020

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Gedecentraliseerd en transparant versiebeheer aan de hand van blockchain principes en IPFS: een praktische toepassing voor open source bedrijven.

Michiel Schoofs

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

Promotor:
Karine Samyn
Co-promotor:
Maurice Dalderup

Instelling: —

Academiejaar: 2019-2020

Tweede examenperiode

Woord vooraf

Samenvatting

ple

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (Pollefliet, 2011):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1 Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgelijnd zijn. Doelgroepen als "bedrijven," "KMO's," systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2 Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specifiëren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3 Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

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 BibLaTeX-formaat (een tekstbestand). Als je expliciet naar de auteur verwijst in de zin, gebruik je `\textcite{}`. Soms wil je de auteur niet expliciet vernoemen, dan gebruik je `\autocite{}`. In de volgende paragraaf een voorbeeld van elk.

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

2.1 Versiebeheer

2.1.1 Inleiding

Versiebeheer is een belangrijk concept binnen softwareontwikkeling. Zo waren er in totaal 100 miljoen projecten op het populaire versiebeheer platform GitHub (in 2018) (Warner, 2018). GitHub (sinds 2018 overgenomen door Microsoft) is echter niet de enige speler op de markt. Zo is er ook nog Code Commit van Amazon en GitLab. Veel bedrijven en oplossingen spelen dus in op de behoefte voor een duidelijk en efficiënt versiebeheer systeem. Toch kan men stilstaan bij de vraag: Welke behoefte lossen deze systemen op?

Stel volgende scenario voor: Alice en Bob zijn aangenomen om te werken voor Bedrijf X. Hun eerste taak is een website ontwikkelen. Ze leggen samen alle vereisten vast, bespreken de verschillende technologieën en gaan aan de slag. Op het einde van de eerste dag hebben ze elk een verschillende pagina gemaakt en deze willen ze graag met elkaar delen.Dit kan door bijvoorbeeld via mail de bestanden door te sturen. Een andere mogelijkheid is de bestanden via fysieke hardware zoals een USB-Stick aan elkaar te geven. Het nadeel is dat de code op twee verschillende plaatsen verspreid zit. Als Bob de code die hij heeft geschreven kwijt raken, dan zal deze opnieuw moet worden geschreven. Om dit probleem te voorkomen kan men het project op een centrale server gaan opslaan. Bob en Alice zullen hun veranderingen opslaan op deze centrale server. Zo hebben ze altijd toegang tot elkaars werk.

Deze manier van werken heeft zijn eigen nadelen. Alice kan per ongeluk een bestand overschrijven of een stuk code verwijderen. Tenzij men back-ups heeft is het originele bestand verloren. Om dit probleem te omzeilen wordt er gebruik gemaakt van het concept van **versies**. Elke aanpassing die er gemaakt wordt resulteert in een nieuwe versie van het project. Men kan altijd terugkeren naar een eerdere versie. Als Alice dus het stukje code verwijdert in versie 15 kan men terug gaan naar versie 14.

Loeliger (2012) stelt dat een versiebeheersysteem een middel is om verschillende versie van code te gaan beheren en bijhouden. De auteur onderscheid volgende drie eigenschappen waaraan dergelijke systemen voldoen:

- Er wordt gebruik gemaakt van een centraal Archief. Binnen dit archief worden alle versies van het project bewaard en bijgewerkt.
- Het centraal archief geeft toegang tot eerdere versies van het project.
- Alle veranderingen die worden aangebracht aan het archief worden genoteerd in een centraal logboek.

Versiebeheer is geen nieuw concept. Er zijn zoals eerder aangehaald verschillende software oplossingen beschikbaar. Toch zijn er volgens Chacon en Straub (2014) drie grote categorieën (zie 2.1 voor een grafische weergave):

- Lokale versiebeheersystemen: het centraal archief waar de veranderingen in worden bewaard staat op een lokale computer. Het grootste voordeel is dat een lokaal systeem zeer makkelijk te onderhouden is. Het is eveneens eenvoudig op te stellen.

Toch is het niet geschikt om bestanden met elkaar te delen of samen aan bestanden te werken. Een gekend voorbeeld is RCS (Revision Control System) - zie 2.1.2 -.

- CVCS: Om samen te kunnen werken aan dezelfde bestanden kan een CVCS (Centralised Version Control System) worden gebruikt. In plaats van het archief lokaal bij te houden wordt er gebruik gemaakt van een centrale server. Bestanden worden vervolgens lokaal gekopieerd. Als er veranderingen worden aangebracht zullen deze worden doorgestuurd naar de server.Doordat men verplicht is om de bestanden op een centrale plaats af te halen, kan men deze gaan afschermen. Zo kan men toegang beperken tot enkel de nodige bestanden per gebruiker.

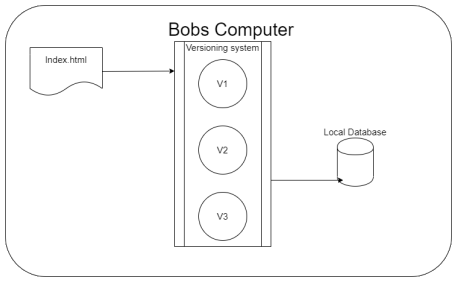
  Een neveneffect van alles centraal te gaan beheren is het zogenaamde *single point of failure(SPOF)* probleem. Een SPOF is een onderdeel van een systeem dat mocht het uitvallen heel het systeem tot een halt roept. Met andere woorden valt het centraal archief weg heeft niemand nog toegang tot het project. Een mogelijke oplossing voor dit probleem is redundantie. Dit betekent het aanbieden van kopieën. (Microsystems, 2007)

- DVCS: Om het SPOF probleem te voorkomen kan men kopieën maken van het centraal archief. Deze kopieën kunnen vervolgens worden verspreid over verschillende computers. Dit is het uitgangspunt van DVCS (Distributed version control System). Elke gebruiker heeft een lokale kopie van de centrale server. De veranderingen aan de bestanden worden eerst aangebracht in het lokaal archief en vervolgens gesynchroniseerd met de centrale variant.

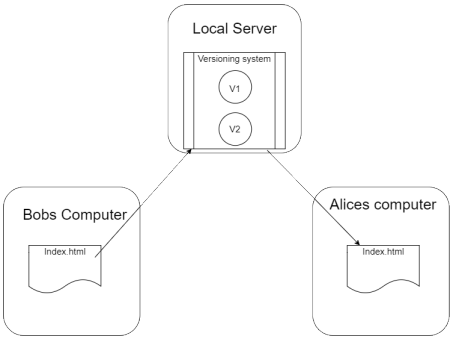
  Mocht het centraal aanspreekpunt niet beschikbaar zijn is dit geen probleem. Elke gebruiker heeft immers een volledige back-up van het volledige project. In theorie kan de gebruiker zelfs optreden als nieuwe centrale server.

2.1.2 RCS

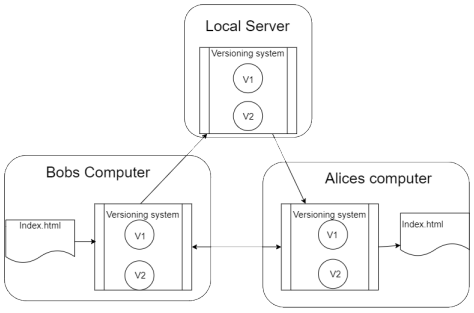
RCS is een klassiek voorbeeld van een lokaal versiebeheer systeem. Het is geschreven en geformaliseerd door Tichy (1985) in een artikel gepubliceerd aan de universiteit van Purdue in de verenigde staten. GNU -een besturingssysteem dat zeer actief is op het gebied van vrije en democratische software- nam het project over als vervanging voor het toenmalige CSSC (Youngman, 2016). CSSC was een gratis verkrijgbare variant van het SCCS (Source code control system) dat eigendom was van Bell Labs. SCCS is een versiebeheer systeem geschreven door Rochkind (1975) voor Unix systemen. CSSC en SCCS waren echter niet de enige systemen voor RCS. Zo was er ook nog CA-Panvalet een gepatenteerde oplossing voor Mainframe computers en nog enkele andere.Toch is het interessant om stil te staan bij RCS ten opzichte van eerdere systemen. Dit is omdat veel van de concepten die waarvan het systeem gebruik maakt zijn nog steeds terug te vinden in moderne versiebeheer systemen (zoals GIT). RCS kadert eveneens binnen de algemene ideologie van deze bachelorproef zo is het volledig open source en wordt het nog steeds op vrijwillige basis onderhouden. Voor deze sectie wordt voornamelijk gesteund op de



(a) Overzicht van de structuur van een Lokale VCS.



(b) Overzicht van de structuur van een CVCS.



(c) Overzicht van de structuur van een DVCS.

Figuur 2.1: Overzicht van de drie types van VCS zoals aangegeven door Chacon en Straub (2014).

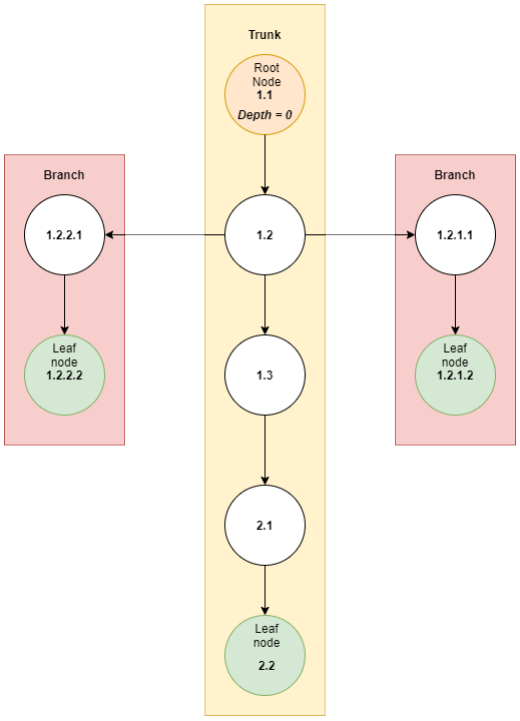
originele paper door Tichy (1985).

Een eerste concept waar de software gebruik van maakt is het concept van een boomstructuur . Een mooi voorbeeld van dit soort structuur is een stamboom. Voor onderstaande definities wordt gesteund op Lievens (2019). Een boomstructuur bevat een collectie van **toppen** *(in het engels ook wel Nodes genoemd)*. Helemaal aan het begin van de boom ligt een top die we ook wel bestempelen als **wortel** *(Root)*. Een andere speciale top is het **blad** *(leaf)* dit is een top die geen kinderen heeft. Alle tussenliggende toppen worden dan ook bestempeld als intermediair. Elke top heeft een arbitrair aantal kinderen (synoniem van opvolgers)die op hun beurt opnieuw een wortel zijn voor een deelboom. Tot slot speelt binnen RCS ook het concept van **diepte** een rol. De diepte van de wortel is nul (d=0) en elk kind heeft als diepte:

$$d_{kind} = d_{ouder} + 1 \tag{2.1}$$

Al deze concepten worden ook nog eens grafisch verduidelijkt in de grafiek **??**.

RCS gebruikt een boomstructuur om de verschillende versies te gaan voorstellen. Stel bijvoorbeeld dat Bob en Alice bezig zijn aan hun hoofdpagina. Ze willen graag hun verschillende versies gaan delen met elkaar. Ze beschikken over één centrale computer waar ze beiden toegang toe hebben via een terminal. Bob maakt een initiële versie van de



Figuur 2.2: Een overzicht van alle concepten binnen een boomstructuur waar RCS van gebruik maakt.

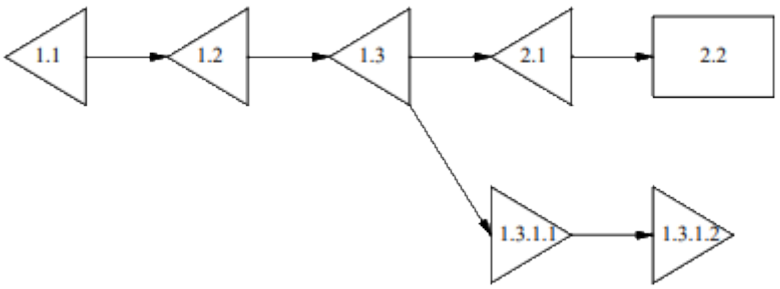
pagina en gebruikt vervolgens het commando voor het **inchecken** (`ci homepage.html`). Er wordt gevraagd voor een korte beschrijving van het project en vervolgens wordt de extensie *.v* toegevoegd. Het originele bestand zal standaard ook verwijderd worden. Het equivalent voor inchecken bij GIT is het `git Push` commando.

Het bestand krijgt ook een **versie nummer** onder de vorm $x_1.x_2$ waarbij x_1 voor een grote verandering staat (*release*) en x_2 voor een kleine verandering (*level*). Deze manier van versies te gaan labelen wordt nog steeds gebruikt. Er zijn ook andere mogelijkheden zoals *Semantic versioning* die worden gebruikt. De versie die bob als eerste online heeft gezet krijgt het nummer 1.1. Elke volgende check-in van het bestand zal het level (x_2) met één gaan verhogen. De volgende versie wordt dus 1.2. Het release nummer (x_1) wordt manueel verhoogd doormiddel van de *-r* optie bij check-in. Bij branches is er ook nog spraken van x_3 en x_4 zie 2.1.2. Dit concept bestaat in Git onder de vorm van *tags* (2.1.4).

Er is nu een bestand onder de vorm `homepage.html.v` maar hoe kan Alice nu dit bestand gaan aanpassen? Alice gaat het bestand moeten **uitchecken** door middel van het commando `co homepage.html`. Op die manier kan de nieuwste versie van het bestand opgevraagd worden. Vervolgens worden de wijzigingen aangebracht en gaat het bestand door Alice weer worden ingecheckt. Met de *-r* optie kunnen we een specifieke versie gaan ophalen. Het equivalent van `co` binnen git is het concept van `pullen`.

Hoe weet RCS echter wat het verschil is tussen versie 1.1 en 1.2? Een eerste naïeve oplossing zou zijn om alle versies van het bestand afzonderlijk te gaan bijhouden. Dit vraagt echter veel opslagruimte. RCS gebruikte voor dit probleem het concept van **deltas**. Een delta is een bestand dat bijhoud welke lijnen van je bestand concreet veranderd zijn tussen de versies. Heeft men dus één lijn verwijderd in een bestand zal de delta maar één regel gaan opnemen [1]. Er zijn twee types van deltas: **voorwaardse deltas**en **achterwaardse deltas**. Bij het in-checken van een nieuwe versie zal de vorige versie worden vervangen door een achterwaardse delta. Zit men momenteel op versie 1.3 en vraagt men versie 1.2 dan zal de achterwaardse delta van versie 1.2 worden toegepast op versie 1.3. Voorwaardse deltas komen aanbod in het gedeelte over branching (2.1.2).Dit concept wordt ook nog eens verduidelijkt in onderstaande grafiek **??**

Door dit principe van inchecken en uitchecken hebben we dus al een werkend systeem. Er is echter wel nog een probleem stel dat Bob en Alice samen versie 1.2 hebben uitgecheckt, ze brengen alle twee wijzigingen aan en willen in-checken. Wie krijgt voorrang? Dit probleem wordt opgelost door het concept van **sloten**(engels=lock). Op het moment dat Bob zijn versie gaat uitchecken kan hij deze versleutelen (door middel van de *-l* optie bij het co commando). Hierdoor kan niemand anders dan Bob een nieuwe versie gaan in-checken [2]. Het bestand ligt dus vast tot Bob het slot vrijgeeft door een nieuwe versie te gaan publiceren [3]. Deze manier van werken heeft duidelijk een groot aantal nadelen. Alice is verplicht om te wachten op Bob zijn nieuwe versie alvorens ze veranderingen kan maken. Git zal dit probleem anders gaan aanpakken door middel van het introduceren van *Merges*.De veranderingen die Alice aanbrengt en de verandering van bob zouden door merges worden samengevoegd tot één nieuwe versie.



Figuur 2.3: Een voorbeeld van deltas.De Trunk bevat een series van achterwaardse deltas terwijl alle branches enkel voorwaardse deltas bevatten. Grafiek afkomstig uit Tichy (1985)

2.1.3 RCS

2.1.4 GIT

[1] De delta wordt opgebouwd aan de hand van het GNU commando diff https://www.gnu.org/software/diffutils/

[2] Andere gebruikers kunnen echter wel nog de versleutelde versie gaan bekijken

[3] In sommige gevallen kan het slot ook worden 'geforceerd' mocht Bob bijvoorbeeld ziek vallen

3. Methodologie

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, consectetuer 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. 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

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 consectetuer 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.

A. 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 Introductie

De onderzoeksvraag waaruit wordt vertrokken luidt als volgt: **"Hoe kunnen we doormiddel van IPFS en Blockchain technologie versiebeheer van een Client-Server architectuur naar een gedecentraliseerde Peer-to-peer model overzetten?"** Binnen deze onderzoeksvraag zijn er vier begrippen:

- **Versiebeheer**: Git -een grote speler op het gebied van Versiebeheer- hanteert volgende definitie van het begrip “Versiebeheer is het systeem waarin veranderingen in een bestand of groep van bestanden over de tijd wordt bijgehouden, zodat je later specifieke versies kan opvragen.” Deze definitie werd gepubliceerd in het boek *Pro git* (Chacon & Straub, 2014).

- **IPFS**: Interplanetary File System of (IPFS)werd in de paper “IPFS - Content Addressed, Versioned, P2P File System” geïntroduceerd door Benet (2014). Hierin stelde hij zijn technologie voor als een peer-to-peer gedistribueerd bestandssysteem waarin alle computers werken met dezelfde bestandsindeling. Hiermee wordt bedoeld dat bestanden kunnen worden opgedeeld in verschillende delen (*ook wel*

'Shards' genoemd) en vervolgens worden opgeslagen op verschillende computers op een gezamenlijk netwerk. Vervolgens kunnen bestanden worden opgevraagd door middel van dit gezamenlijk netwerk aan te spreken. Er is dus geen gecentraliseerd aanspreekpunt.

- **Blockchain**: Blockchain is een manier om data op te slaan aan de hand van blokken. Deze blokken bevatten verschillende gegevens. Deze gegevens worden omgezet door middel van wiskundige functies die ook wel hashfuncties worden genoemd. Door de blokken onderling aan elkaar te koppelen en wiskundige functies te gebruiken bij het verifiëren van de integriteit van de blokken ontstaat er een veilige en volledig gedecentraliseerde manier van dataopslag.

- **P2P**: Tot slot is er ook het concept van Peer-to-peer (courant afgekort als P2P). Voor een gangbare definitie kan gebruik gemaakt worden van de werken van Schollmeier (2001) en IAB en Gonzalo (2009). Hierbij wordt gesteld dat P2P bestaat uit verschillende computers (nodes) die onderling met elkaar verbonden zijn. Deze nodes vervullen daarbij de rol van zowel server als client (zogenaamde Servents). Hierdoor kunnen de verschillende nodes diensten en data aan elkaar opvragen en delen zonder een centraal aanspreekpunt (client-server architectuur). Dit is dan ook het achterliggende principe van IPFS. Blockchain is een manier om gegevens en gedragsintegriteit (als gedefinieerd in (Drescher, 2017)) te waarborgen zonder centrale autoriteit binnen P2P netwerken.

Versiebeheer wordt vaak uitbesteed aan derden of zelf gedaan aan de hand van een centrale server. Het nadeel hiervan is dat er één centrale plek is waar het kan mislopen. Stel bijvoorbeeld dat de centrale server gegevens verliest is men alles kwijt. Een ander probleem is dat er binnen versiebeheer een aantal monopolies ontstaan waaronder Microsoft die GitHub kocht in 2018. Dit staat haaks op de open source beweging die streeft naar een transparante en democratische manier van software ontwikkeling. Door het introduceren van de bovengenoemde technologieën kunnen zowel de bestanden als de nodige informatie voor versiebeheer worden verspreid waardoor er geen centraal punt is en er ook geen commercieel bedrijf bij betrokken is.

De doelstelling van dit onderzoek is om versiebeheer te decentraliseren. Daaronder wordt verstaan overstappen van de klassieke server-client architectuur zoals github (of een lokaal gehost versiebeheer systeem) naar een P2P netwerk. Voor de bestanden wordt gebruik gemaakt van de reeds bestaande IPFS technologie. Hierbij zal een blockchain oplossing worden ontwikkeld om het geheel te ondersteunen (metadata, manifest bestanden,...). -Zie ook A.3 Methodologie.-

Om de onderzoeksvraag volledig te beantwoorden kan deze nog verder opgesplitst worden in verschillende deelvragen. Deze vragen komen dan ook chronologisch aan bod om

uiteindelijk tot een werkend systeem te bekomen. De verschillende deelvragen die worden behandeld zijn:

- Wat zijn de problemen van versiebeheer binnen softwareontwikkeling en hoe worden deze aangepakt door versiebeheer systemen zoals GIT?
- Waarom zouden we van gecentraliseerde (server-client architectuur) versiebeheer systemen overstappen naar een gedecentraliseerde variant?
- Wat zijn de eigenschappen en valkuilen van gedecentraliseerde (P2P) netwerken?
- Hoe gaan protocollen zoals Gnutella en BitTorrent te werk voor Peer-to-peer filesharing?
- Wat is IPFS en hoe vergelijkt het met andere gedecentraliseerde filesharing protocollen?
- Op welke manier biedt IPFS een meerwaarde ten opzichte van klassieke versiebeheer systemen?
- Wat zijn de basisprincipes van Blockchain?
- Hoe kunnen data veilig en integer worden bewaard op een Blockchain netwerk?
- Welke meerwaarde kan blockchain bieden binnen de context van Peer-to-peer netwerken?
- Wat zijn smartcontracts en hoe kunnen ze een meerwaarde bieden binnen blockchain oplossingen?
- Op welke wijze kunnen we IPFS en blockchain combineren tot een werkzaam versiebeheer systeem?

A.2 State-of-the-art

De manier van werken is gebaseerd op het artikel “Decentralized document version control using ethereum blockchain and IPFS.” (Nizamuddin e.a., 2019) In het onderzoek wordt er gebruik gemaakt van smart contracts. Dit is in essentie code die zal uitgevoerd worden als aan bepaalde voorwaarden wordt voldaan. Deze smart contracts worden gebruikt om de verschillende aspecten van versiebeheer en data vast te leggen en uit te voeren. Voor de bestanden binnen het project wordt gekozen voor IPFS om op een gedecentraliseerde manier deze te kunnen opslaan.

De paper vormt een zeer goede aanzet en ook de werkmethode is uitvoerig beschreven. Toch blijft het zeer abstract. Belangrijke aspecten van versiebeheer worden kort of niet aangehaald waaronder “cloning”, “merging” of “branching”. In de bovengenoemde paper wordt een sterke focus op Ethereum gelegd, ontwikkeling bovenop deze blockchain interpretatie brengt echter significante overhead met zich mee. Zo is de snelheid van het systeem afhankelijk van de capaciteit en belasting van het netwerk op het gegeven moment.

Deze bachelorproef legt de focus op het ontwikkelen van een concrete toepassing. Ook de meer complexe en technische problemen zullen worden behandeld. De algemene principes van blockchain zullen vrijer worden geïmplementeerd en op een lokaal netwerk van enkele computer worden verspreid. In plaats van een grotere architectuur en implementatie te gebruiken

A.3 Methodologie

Er wordt vertrokken vanuit een literatuurstudie om de verschillende elementen van versiebeheer en de reeds bestaande technologieën te verkennen. Vervolgens komen de aspecten van blockchain en IPFS aan bod door middel van een demo waarin wordt gebruik gemaakt van een Word document met verschillende versies.

Tot slot worden de verschillende aspecten van versiebeheer aan de hand van een demo-applicatie geïllustreerd. Hiervoor zijn er drie hypothetische gebruikers: Alice, Bob en Carol die samen een T-shirt webshop ontwikkelen. Ze zullen hiervoor gebruiken maken van ASP.Net en Visual Studio. Binnen hun ontwikkelingsproces zullen ze een aantal gekende problemen tegenkomen waaronder “merge conflicten” en verschillende “branches”.

Bij elk van die problemen wordt er gekeken naar hoe Git -een klassiek versiebeheer systeem- dit oplost en hoe er een oplossing kan voorzien worden vanuit de voorgestelde gedistribueerde blockchain benadering. Voor het opstellen van de blockchain wordt gebruik gemaakt van C# en Nethereum (Juan Blanco, 2020). Aangezien er wordt gesteund op de IPFS API wordt er gebruik gemaakt van de open source bibliotheek net-ipfs-client-http geschreven door Richard Schneider (2019). De bedoeling is om op het einde van de bachelorproef tot een werkend prototype te komen dat gebruikt kan worden voor verschillende doeleinden.

A.4 Verwachte resultaten

Het eindresultaat van de Bachelorproef is om op een onderbouwde manier een prototype aan te reiken om op gedecentraliseerde wijze aan versie beheer te gaan doen. De voorgestelde werkwijze wordt grondig vergeleken met Git op de volgende twee punten:

- Snelheid van een transactie: hoe lang duurt het om bewerkingen zoals pull requests en branching toe te passen op een project en/of branch?
- Performantie qua geheugengebruik: hoe efficiënt wordt er binnen de algoritmen van de oplossing omgesprongen met geheugengebruik? Hiermee wordt zowel het extern geheugen (Hardeschijf, SSD) als het werkgeheugen bedoelt.

De verwachting is dat de implementatiesnelheid lager zal zijn dan met de klassieke Git-systemen, omdat blockchain van nature vrij omslachtig en intensief is. De performantie van het geheugensysteem zal eveneens slechter scoren ten opzichte van Git. De blockchain implementatie waarborgt echter een hogere mate van data integriteit .

A.5 Verwachte conclusies

Gedecentraliseerd versiebeheer door middel van Blockchain en IPFS is zeker technisch mogelijk. De voordelen die het biedt zijn niet alleen zuiver ideologisch.De inherente garantie op data integriteit en het weghalen van een centraal "*point of failure*" is interessant voor grote bedrijven met een groot aantal aan verschillende projecten. Er zijn echter een aantal nadelen verbonden waaronder de omslachtige procedure en het intensief gebruik van computertechnische middelen. Dit maakt het voor kleine bedrijven minder interessant.

Bibliografie

Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System.

Chacon, S. & Straub, B. (2014). *Pro Git* (2nd). Berkely, CA, USA, Apress.

Creeger, M. (2009). CTO Roundtable: Cloud Computing. *Communications of the ACM*, *52*(8), 50–56.

Drescher, D. (2017). *Blockchain Basics : A Non-Technical Introduction in 25 Steps*. New York, APRESS. http://blockchain-basics.com/

IAB & Gonzalo, C. (2009). Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability. RFC Editor. https://doi.org/10.17487/RFC5694

Juan Blanco. (2020, januari 27). *Nethereum* (Versie 3.6.0). https://github.com/Nethereum/Nethereum

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Redwood City, CA, USA, Addison Wesley Longman Publishing Co., Inc.

Lievens, S. (2019, februari 5). *Probleemoplossend Denken II Lesnota's*.

Loeliger, J. (2012, september 1). *Version Control with Git*. O'Reilly UK Ltd. https://books.google.be/books?hl=nl&lr=&id=aM7-Oxo3qdQC&oi=fnd&pg=PR3&dq=Version+control&ots=39BeLFUfqd&sig=V5WFl33nbxhbMH1r97EwxMfmdqs#v=onepage&q&f=false

Microsystems, S. (2007, maart). *Sun Java System Directory Server Enterprise Edition 6.0 Deployment Planning Guide* (S. Microsystems, Red.). Verkregen 29 februari 2020, van https://docs.oracle.com/cd/E19693-01/819-0992/fjdch/index.html

Nizamuddin, N., Salah, K., Azad, M. A., Arshad, J. & Rehman, M. (2019). Decentralized document version control using ethereum blockchain and IPFS. *Computers & Electrical Engineering*, *76*, 183–197. https://doi.org/10.1016/j.compeleceng.2019.03.014

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Gent, Academia Press.

Richard Schneider. (2019, augustus 30). *net-ipfs-http-client* (Versie 0.33.0). https://github.com/richardschneider/net-ipfs-http-client

Rochkind, M. J. (1975). The source code control system. *IEEE Transactions on Software Engineering*, *SE-1*(4), 364–370. https://doi.org/10.1109/tse.1975.6312866

Schollmeier, R. (2001). A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. https://doi.org/10.1109/P2P.2001.990434

Tichy, W. F. (1985). RCS – A System for Version Control. *Software: Practice and Experience*, *15*(7), 637–654. https://doi.org/10.1002/spe.4380150703

Warner, J. (2018, oktober 8). *Thank you for 100 million repositories* (J. Warner, Red.). https://github.blog/2018-11-08-100m-repos/

Youngman, J. (2016, juni 11). *GNU CSSC* (J. Youngman, Red.). https://www.gnu.org/software/cssc/