HOGESCHOOL ROTTERDAM

BACHELOR SCRIPTIE

Schaalbaarheid Developers.nl

Auteur:
Kaj de Munter
0911825

Begeleiders:
Tanja Ubert
Judith Lemmens

24/09/2019

v0.2

DEVELOPERS.NL

HOGESCHOOL ROTTERDAM

BACHELOR SCRIPTIE

Schaalbaarheid Developers.nl

Auteur:
Kaj de Munter
0911825@hr.nl
k.demunter@developers.nl
06-81019142

Schoolbegeleiders:
Tanja Ubert
t.ubert@hr.nl
Judith Lemmens
j.h.i.lemmens@hr.nl

Bedrijfsbegeleiders:
Maarten de Boer
m.deboer@developers.nl
Jelle van de Haterd
j.vandehaterd@developers.nl

Een scriptie ingediend ter voldoening aan de vereiste competenties voor de opleiding Informatica

Communicatie, Media, en Informatietechnologie

24/09/2019

v0.2

HOGESCHOOL ROTTERDAM

Abstract

Communicatie, Media, en Informatietechnologie

Informatica

Schaalbaarheid Developers.nl

door Kaj de Munter

Samenvatting van de scriptie die ik als laatst pas ga schrijven... Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Voorwoord

Bedankt aan iedereen die mij koffie heeft gebracht afgelopen periode... Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Inhoud

Abstract **iii**

Voorwoord **v**

Figurenlijst **ix**

Tabellenlijst **xi**

Begrippenlijst **xiii**

Lijst van afkortingen **xv**

1 Inleiding **1**
1.1 Aanleiding . . . 1
1.2 Belang . . . 1
1.3 Opdrachtgever . . . 2
1.3.1 Core business . . . 2
1.3.2 Geschiedenis . . . 2
1.3.3 Werkomgeving . . . 3
1.3.4 Taken . . . 4
1.4 Doelstelling . . . 4
1.5 Probleemstelling . . . 4
1.5.1 Hoofd- en Deelvragen . . . 4
1.6 Methodologie . . . 5
1.7 Planning . . . 5
1.8 Leeswijzer . . . 5

2 Theoretisch Kader **7**
2.1 Schaalbaarheid . . . 7
2.1.1 Schaalbaarheid controleren . . . 8
2.1.2 Functionele schaalbaarheid . . . 9
2.2 Onderhoudbaarheid . . . 10
2.3 Architectuur . . . 10

A Frequently Asked Questions **13**
A.1 How do I change the colors of links? . . . 13

Literatuurlijst **15**

Figurenlijst

Tabellenlijst

1.1 Planning 5

Begrippenlijst

Kubernetes	Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications [1].
Docker	Docker is a tool designed to make it easier to create, deploy, and run applications by using containers [2].
Ansible	Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs [3].
Docker Swarm	Docker Swarm provides native clustering functionality for Docker containers, which lets you turn a group of Docker engines into a single, virtual Docker engine [4].
Infrastructure as Code	Infrastructure as code describes the idea of using a high-level programming language to control IT systems [5].
Ansible Tower	Red Hat Ansible Tower helps you scale IT automation, manage complex deployments and speed productivity. Centralize and control your IT infrastructure with a visual dashboard, role-based access control, job scheduling, integrated notifications and graphical inventory management [6].
Chef	Deploy new code faster and more efficiently. Automate infrastructure and applications [7].
Puppet	Powerful infrastructure automation and delivery [8].
Terraform	Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions [9].
Cloud computing providers	Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [10]. Er bestaan veel cloud computing providers, waaronder bijvoorbeeld: • Amazon web services • DigitalOcean • Microsoft Azure • Google Cloud Platform

12-factor app	Een methodologie voor het bouwen van Software as a Service (SaaS) applications [11].
The Open Group Architecture Framework	A generic framework to build different IT architectures frameworks [12].
4+1 architectural view model	A model for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views [13].

Lijst van afkortingen

MI	**M**aintainability **I**ndex
CI	**C**ontinuous **I**ntegration
CD	**C**ontinuous **D**eployment
EMS	**E**mployee **M**anagement **S**ystem
IaC	**I**nfrastructure **A**s **C**ode
K8s	**K**ubernete**s**
SaaS	**S**oftware **a**s **a** **S**ervice
TOGAF	**T**he **O**pen **G**roup **A**rchitecture **F**ramework

Hoofdstuk 1

Inleiding

1.1 Aanleiding

"Een visitekaartje voor het bedrijf". Dat is het uitgangspunt wanneer interne software bij Developers.nl wordt ontwikkeld. Niet alleen qua uiterlijk, maar ook van binnen moet de code, de infrastructuur en de werkmethode in topconditie zijn. Dit heeft te maken met het feit dat de code uit de website en infrastructuur van Developers.nl open-source wordt gemaakt gedurende deze stage.

Dit betekent dat er veel onderzoek moet worden uitgevoerd op de kwaliteit van de huidige website. Met name de infrastructuur, het "Continuous Integration/Continuous Deployment" (CI/CD) en het horizontaal en verticaal schalen hebben nog veel ruimte voor verbetering.

Bovendien zijn er meerdere interne systemen dan alleen de website. Het onderhouden van (binnenkort) 5 systemen vereist veel tijd en moeite, dus moet dit makkelijker worden.

1.2 Belang

Het open-source maken van de website betekent dat elke potentiële klant en/of nieuwe medewerker de mogelijkheid heeft om te bekijken wat Developers.nl qua kennis in huis heeft. Het is dus van hoog belang dat de kwaliteit gewaarborgd wordt, en dat er zo veel mogelijk nieuwe en opkomende technieken worden gebruikt.

Bovendien heeft Wheeler [14] geconcludeerd dat open-source software voordelen heeft als:

- Betere beveiliging
- Betere betrouwbaarheid
- Betere prestaties
- Betere schaalbaarheid
- Mindere onderhoudskosten

Ook al is de website op eerste gezicht van de buitenkant vrij eenvoudig, er moet (om indruk te wekken op potentiële klanten en nieuwe medewerkers) onder water een stevige applicatie draaien dat eigenlijk iets "te complex" is. Kortom, er is nog veel te verbeteren en toe te voegen.

Het onderhouden van (binnenkort) 5 interne systemen betekent dat er veel werk dubbel wordt uitgevoerd. Dit kost geld, en kan beter.

1.3 Opdrachtgever

Deze scriptie is geschreven in opdracht van:

Developers.nl
Stadionweg 57C
3077AS Rotterdam
info@developers.nl
010-3035929

1.3.1 Core business

Developers.nl neemt software ontwikkelaars in dienst. De ontwikkelaars die worden aangenomen zullen voornamelijk gespecialiseerd zijn in PHP, Python, Java of front-end. Ze worden uitgezet naar een klant (een extern bedrijf) die naar een ontwikkelaar zoekt. Developers.nl kiest hier voor de beste ontwikkelaar voor de taak en zal deze inzetten bij een klant. De opdrachten van de ontwikkelaars zijn op locatie van de klant en duren voornamelijk langer dan een jaar, maar op uitzondering zijn er ook kortere opdrachten. Zodra de ontwikkelaar klaar is met zijn of haar taak zal Developers.nl zo snel mogelijk een nieuwe opdracht toewijzen [15]. Concreet zegt het positioneringsprofiel [16]: "Detachering van developers die software applicaties bouwen voor verschillende klanten."

1.3.2 Geschiedenis

Mark Smit heeft altijd al in de detacheringshoek gewerkt. Hieruit is de inspiratie begonnen om een detacheringsbedrijf voor zichzelf op te richten in 2012. Het bedrijf heette "JC Capacity" en was onderdeel van "JC Groep". Mark Smit was mede eigenaar met 2 anderen binnen JC Groep. Er was op dit moment alleen nog een kleine groep PHP ontwikkelaars, totdat in 2013 een aantal Java en Python ontwikkelaars zijn aangenomen. In februari 2014 is Mark Smit samen met een andere eigenaar binnen JC Group als JC Capacity afgesplitst. Dit gebeurde in verband met meningsverschillen over welke kant het bedrijf op moest doordat JC Groep meer op de zorg gefocust was. Gelijk hierna is een naamsverandering plaatsgevonden en heette het bedrijf "Developers.nl".

Kort na de naamsverandering is het bedrijf exponentieel gegroeid, de teams PHP, Java en Python zijn individueel gegroeid, maar ook is er een nieuwe Front-End tak bijgekomen. vanaf 2015 is heeft de andere eigenaar het bedrijf verlaten en was Mark Smit de enige eigenaar van Developers.nl De naamsverandering en de splitsing van JC Groep heeft ervoor gezorgd dat Developers.nl meer vrijheid had om veranderingen aan te brengen en naamsbekendheid op te krikken.

In 2016 heeft Mark Smit een tweede bedrijf opgericht met de naam "Gemvision". Dit bedrijf vereiste meer van zijn tijd op dan hij had verwacht, en heeft hierdoor vanaf 2018 zijn management taken binnen Developers.nl overgegeven aan Maarten de Boer [15].

1.3.3 Werkomgeving

Cultuur

Developers.nl is een jonge en ambitieuze club mensen met een sterke onderlinge samenhang. Developers is een platte organisatie. Er heerst een sfeer van vrijheid, verantwoordelijkheid en "work hard, play hard". De omgang met elkaar is informeel en persoonlijk, met het gevoel van een vriendenclub. De organisatie wordt gerund vanuit het besef dat plezier in het werk belangrijk is. De dresscode is business casual zodat klanten hetzelfde gevoel hebben van vrijheid en verantwoordelijkheid bij de ontwikkelaars. Op het kantoor luncht iedereen gezamenlijk en wordt er regelmatig geborreld na werktijd. Drinken is inbegrepen en lunch moet zelf meegebracht worden. Volgens het Competing Values Framework van Robert Quinn en Rohrbaug (1983) valt de cultuur onder mensgericht, de medewerkers staan centraal [17].

Er is veel aandacht voor de medewerker als mens en zijn of haar persoonlijke leven, ook buiten werktijd. De open cultuur biedt ruimte voor het inbrengen van ambities, ideeën, vragen en wensen, op welk gebied dan ook. Vanuit het principe van wederkerigheid spant de organisatie zich in om de medewerker zoveel mogelijk te helpen en een goede werkgever te zijn, zodat medewerkers zich inspannen voor hun werkgever, collega's en klanten.

Ontwikkeling staat in alle opzichten centraal en strekt zich uit in twee richtingen: De ontwikkeling van medewerkers, zowel persoonlijk als professioneel, en de ontwikkeling van individuele en collectieve vakinhoudelijke kennis.

Developers.nl faciliteert kennis aan haar eigen ontwikkelaars in de vorm van onder andere "TechNights", kennissessies en gezamenlijke deelname aan conferenties in het buitenland. Het zijn van een kennisorganisatie werkt twee kanten op: klanten met uitdagende opdrachten weten Developers.nl hiervoor te vinden en dat maakt de werksituatie compleet om medewerkers blijvend te boeien.

Klanten ervaren Developers.nl als oprecht geïnteresseerd, persoonlijk, analytisch, flexibel en vlot schakelend. Medewerkers zijn eerlijk en open, communicatief sterk, professioneel, betrouwbaar en intelligent. Het feit dat ze kunnen terugvallen op de collectieve kennis van de groep en dat ze bereid zijn om ook met de klant deze kennis te delen maakt ze extra waardevol. Fieldmanagers zijn erg sterk in het doorgeven van de opdracht naar uitvoerenden [15].

Eigen omgeving

Developers.nl heeft rond de 60 software ontwikkelaars. Deze zijn voornamelijk op een externe opdracht bij ene klant. Elke vrijdag zullen 5 van de zogenaamde "kennisambassadeurs" op kantoor zijn. Dit zijn de meest senior ontwikkelaars per team. Deze zijn dan in staat om stagiairs en/of andere medewerkers persoonlijk te helpen, maar ze zijn altijd bereikbaar via Slack of telefonisch.

De bedrijfsbegeleider voor deze stage is **Maarten de Boer**. Dit is de algemene directeur van Developers.nl en is in 2003 afgestudeerd aan de hogeschool Inholland met Strategic marketing. Aangezien Maarten zelf geen technische kennis heeft is er ook een technische begeleider aangewezen: **Jelle van de Haterd**. Jelle is senior developer, DevOps engineer en kennisambassadeur bij Developers.nl. Hij is in 2006 afgestudeerd op de Hogeschool Rotterdam met als opleiding Grafimediatechnologie [18].

Binnen developers.nl word iedere stagiair behandeld als een daadwerkelijk sterke medewerker die een echte bijdrage aan het bedrijf levert. Het team staat altijd voor hen klaar. Op het moment dat iemand aangeeft dat hij/zij iets nodig heeft (denk aan

bijvoorbeeld een macbook voor het programmeren) zal het team er zo veel mogelijk aan doen dat er een goede oplossing wordt aangeboden. Het team staat open voor feedback en behandelen dit op serieuze wijze.

Voor technische vragen zijn de kennisambassadeurs altijd bereikbaar, bij een vraag zullen zij hun best doen om het zo duidelijk mogelijk over te brengen, en zorgen ervoor dat ik het ook daadwerkelijk begrijp. De kennisambassadeurs kijken code inhoudelijk na en geven hier feedback op [15].

1.3.4 Taken

Tijdens de stageperiode zal de stagiair een leidende rol aannemen in een team van 2 part-time studenten en een stagiair. Er gaat onderzoek worden uitgevoerd over de schaal- en onderhoudbaarheid van de infrastructuur aangaande de interne systemen die Developers.nl beheert. Dit omtrent onderzoek over de huidige situatie, technieken die gebruikt kunnen worden, en het automatiseren hiervan. Door het gehele proces moet er extra rekening gehouden worden met de veiligheid van de gekozen oplossing in verband met de gevoeligheid van een aantal systemen, zoals bijvoorbeeld het Employee Management System (EMS). Uit dit onderzoek zullen een aantal verbeteringen daadwerkelijk worden geïmplementeerd en uiteindelijk een advies worden uitgebracht over eventuele vorderingen.

1.4 Doelstelling

Dit onderzoek heeft tot doel het verkrijgen van kennis en inzicht over de schaal- en onderhoudbaarheid van de interne systemen die Developers.nl beheert om vervolgens deze twee factoren te verbeteren.

1.5 Probleemstelling

Bij Developers.nl zijn niet altijd voldoende interne medewerkers beschikbaar om haar eigen systemen te onderhouden. Dit wil zeggen dat zodra er iets moet worden aangepast aan één van die systemen het onaangenaam lang kan duren voordat dit wordt doorgevoerd. Dit kan grote impact hebben wanneer (bijvoorbeeld) de website van Developers.nl niet meer bereikbaar is.

1.5.1 Hoofd- en Deelvragen

Hoofdvraag

Op welke wijze kan Developers.nl de architectuur van hun huidige systemen beter schaal- en onderhoudbaar maken?

Deelvragen

- Hoe goed is de schaalbaarheid van de huidige websites?
- Wat voor technieken zijn er voor schaal- en onderhoudbaarheid?
- Wat voor verbeteringen ten aanzien van schaal- en onderhoudbaarheid kunnen worden toegepast op de huidige websites?
- Hoe gaan deze verbeteringen geïmplementeerd worden?

- Voldoen de verbeteringen aan de vereiste requirements?

1.6 Methodologie

- het type onderzoek
- de wijze waarop je informatie hebt vergaard
- de kwaliteitseisen van je bedrijf en opdrachtgever
- de wijze waarop je jouw bevindingenvalideert
- de wijze waarop je de geldigheid en betrouwbaarheid van je bronnenhebt gecontroleerd
- een onderbouwde keuze voor je projectmethode

1.7 Planning

TABEL 1.1: Planning

Week	Taak
1, 2	Skelet opzet scriptie, inleiding
3, 4	Theoretisch kader, afbakening
5, 6	Deelvraag: Hoe goed is de schaalbaarheid van de huidige websites?
7, 8	Deelvraag: Wat voor technieken zijn er voor schaal- en onderhoudbaarheid?
9, 10	Deelvraag: Wat voor verbeteringen ten aanzien van schaal- en onderhoudbaarheid kunnen wor
11, 12	Deelvraag: Hoe gaan deze verbeteringen geïmplementeerd worden?
13 – 17	Praktijk implementatie
18 – 20	Deelvraag: Voldoen de verbeteringen aan de vereiste requirements?en conclusie

1.8 Leeswijzer

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut.

Hoofdstuk 2

Theoretisch Kader

In dit hoofdstuk worden drie belangrijk begrippen uit de onderzoeksvraag behandeld. Er wordt een literatuuronderzoek gedaan naar de bestaande definities van schaalbaarheid, onderhoudbaarheid en architectuur met betrekking tot software. Hierna wordt het begrip afgebakend tot een concrete definitie waar het onderzoek op terug kan vallen.

2.1 Schaalbaarheid

Mark D. Hill heeft in 1990 onderzoek gedaan naar een concrete definitie naar schaalbaarheid [19]. In zijn onderzoek concludeert hij het volgende:

> *I examined aspects of scalability, but did not find a useful, rigorous definition of it. Without such a definition, I assert that calling a system 'scalable' is about as useful as calling it 'modern'. I encourage the technical community to either rigorously define scalability or stop using it to describe systems.*

Ondertussen zijn er meerdere pogingen gedaan om schaalbaarheid te definiëren. zo zijn L. Duboc, D. S. Rosenblum en T. Wicks op deze conclusie ingegaan en hebben een poging gedaan om een framework te creëren voor karakterisering en analyse van software schaalbaarheid [20]. Zij definiëren schaalbaarheid als:

> *quality of software systems characterized by the causal impact that scaling aspects of the system environment and design have on certain measured system qualities as these aspects are varied over expected operational ranges*

In een onderzoek over de kenmerken van schaalbaarheid en de impact op prestatie heeft A. B. Bondy [21] schaalbaarheid verdeeld in een aantal verschillende aspecten, waaronder:

- **Structural scalability** (het vermogen van een systeem om uit te breiden in een gekozen dimensie zonder ingrijpende wijzigingen in de architectuur)
- **Load scalability** (het vermogen van een systeem om elegant te presteren naarmate het aangeboden verkeer toeneemt)
- **Space scalability** (het geheugenvereiste groeit niet naar "ondraaglijke niveaus" naarmate het aantal items toeneemt)
- **Space-time scalability** (het systeem blijft naar verwachtingen functioneren naarmate het aantal items dat het omvat toeneemt)

Bondy definieert schaalbaarheid als het vermogen van een systeem om een toenemend aantal elementen, objecten en werk gracieus te verwerken en / of vatbaar te zijn voor uitbreiding.

H. El-Rewini en M. Abd-El-Barr noemen in het boek Advanced computer architecture and parallel processing [22] ook een aantal "onconventionele" definities:

- **Size scalability** (Meet de maximale hoeveelheid processors dat een systeem kan accommoderen)
- **Application scalability** (de mogelijkheid om applicatiesoftware te draaien met verbeterde prestaties op een opgeschaalde versie van het systeem)
- **Generation scalability** (de mogelijkheid om op te schalen door het gebruik van de volgende generatie (snelle) componenten)
- **Heterogeneous scalability** (het vermogen van een systeem om op te schalen met behulp van hardware- en softwarecomponenten die door verschillende leveranciers zijn gemaakt)

C.B. Weinstock en J. B. Goodenough hebben een algemeen onderzoek uitgevoerd naar schaalbaarheid [23]. Zij noemen in hun conclusie dat er voornamelijk twee betekenissen van het woord schaalbaarheid zijn:

1. De mogelijkheid om met verhoogde werkdruk om te gaan (zonder extra resources aan een systeem toe te voegen).
2. De mogelijkheid om met verhoogde werkdruk om te gaan door herhaaldelijk een kosteneffectieve strategie toe te passen om de mogelijkheden van een systeem uit te breiden.

Het valt op dat een concrete definitie van schaalbaarheid alleen duidelijk te definiëren is wanneer het in meerdere verschillende soorten is opgesplitst. Daarom zal in dit onderzoek vanaf dit punt altijd worden gespecificeerd welke soort schaalbaarheid het betreft. In dit onderzoek wordt vooral de focus gelegd op de "structural scalability" en "load scalability" uit [21]. "application scalability" uit [22] heeft veel overlapping met de load scalability. Omdat load scalability iets generieker is en de twee definities van Weinstock en Goodenough [23] omvat wordt deze geprefereerd boven application scalability. De overgebleven definities zijn minder relevant voor dit onderzoek aangezien ze te maken hebben met hardware, of niet volledig toepasselijk is op de architectuur.

2.1.1 Schaalbaarheid controleren

In het onderzoek van Weinstock en Goodenough [23] zijn een aantal manieren genoemd om de schaalbaarheid van een systeem te controleren.

Knelpunten

Deze knelpunten hebben vooral te maken de eerste betekenis van Weinstock en Goodenough. Er moet gecontroleerd worden op de toenemende administratieve werkdruk, de "hard-coded" limieten op capaciteit, de user interface en de complexiteitsgraad van algoritmen.

Onthullen van schaalbaarheids-aannames

Dit gaat over het onderzoeken hoe de uitbreiding van een systeem nieuwe problemen kan onthullen. Zodra een systeem zich uitbreidt is er een grotere kans op errors

in de systeemconfiguratie, "zeldzame" errors komen vaker voor, is het belangrijk dat een probleem in het systeem gelokaliseerd blijft, en kan het een stuk complexer en lastiger worden om het systeem te begrijpen.

Schaalstrategieën

Hier wordt bekeken wat voor verschillende tekortkomingen een schaalstrategie heeft. Zodra een systeem een lange termijn leeft kan het zo zijn dat het systeem anders wordt gebruikt, hier moet de strategie op voorbereid zijn. Een strategie moet moet niet te afhankelijk zijn van het feit dat gebruikers weten hoe het systeem wordt geïmplementeerd. Ook moet de strategie voorbereid zijn op de vooruitgang van hardware.

Methoden voor schaalbaarheidsborging

In het onderzoek vertellen Weinstock en Goodenough dat niet echt mogelijk is om te testen of een systeem schaalbaar is. Wel zijn er methoden om de schaalbaarheid te waarborgen, zoals:

- Onderzoek de "performance curves" en karakteriseer deze met een Big O notatie.
- Identificeer mechanismen om knelpunten aan het licht brengen of waar aannames van het schaalbaarheids- ontwerp beginnen te worden geschonden.
- Voer een SWOT analyse uit op de schaalbaarheids-strategie.
  - **S**trengths (de soorten groei waar de strategie voor ontworpen is)
  - **W**eaknesses (de soorten groei waar de strategie niet voor ontworpen is)
  - **O**pportunities (mogelijke veranderingen in werklast of technologie die de strategie goed zou kunnen benutten)
  - **T**hreats (mogelijke veranderingen in de werklast of technologie die de strategie in twijfel zouden kunnen trekken)

Schalen kan op twee verschillende manieren, namelijk horizontaal en verticaal. Horizontaal wilt zeggen dat er geschaald wordt door meerdere machines toe te voegen, terwijl verticaal schalen betekent dat er meer rekenkracht (als bijvoorbeeld een betere CPU of meer RAM) wordt toegevoegd aan een machine. Ook is bij het schaalbaar maken van systemen van belang dat het zo min mogelijk ten koste gaat van prestaties en niet meer kost dan nodig is.

2.1.2 Functionele schaalbaarheid

Wat in de literatuur mist over schaalbaarheid is het schrijven van "schaalbare code". Deze term wordt regelmatig gebruikt in informele bronnen als blogs of vacatures, maar is nooit concreet gedefinieerd. Schaalbare code betekent in welke mate bestaande code moet worden aangepast zodra een nieuwe functionaliteit wordt toegevoegd aan het systeem. In dit onderzoek refereren we naar deze definitie als "functionele schaalbaarheid". Functionele schaalbaarheid ligt heel dicht bij de definitie van onderhoudbaarheid. Deze term zal verder worden uitgelicht in paragraaf 2.2.

2.2 Onderhoudbaarheid

Grubb, P. en Takang, A. A. definiëren in hun boek "Software Maintenance: Concepts And Practice" onderhoudbaarheid als "The discipline concerned with changes related to a software system after delivery" [24]. In 1993 heeft IEEE een "Standard Glossary of Software Engineering Terminology" opgesteld. Deze begrippenlijst definieert onderhoudbaarheid als "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment" [25]. Deze twee definities komen uiteindelijk op hetzelfde neer. Grubb en Takang noemen het in de context van een discipline, terwijl IEEE het als een kwaliteitseigenschap definieert. Ook specificeren Grubb en Takang het feit dat het alleen ná het opleveren gebeurt. In dit onderzoek wordt uitgegaan van de definitie van IEEE.

Grubb en Takang noemen ook een aantal redenen waarom software moet worden onderhouden:

- Ondersteuning van verplichtte upgrades
- Ondersteuning van verzoeken van gebruikers om verbeteringen toe te voegen
- Om toekomstige onderhoudswerkzaamheden te vergemakkelijken

K.K. Aggarwal, Y. Singh en J.K. Chhabra noemen in hun onderzoek een aantal factoren die van invloed zijn op onderhoudbaarheid van software [26]:

- Leesbaarheid van de broncode
- Kwaliteit van de documentatie
- Begrijpelijkheid van software

ISO 25010 [27] definieert onderhoudbaarheid als "The degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements" en verdeeld het in een vijftal kwaliteitseigenschappen.

- Modulariteit
- Herbruikbaarheid
- Analyseerbaarheid
- Wijzigbaarheid
- Testbaarheid

Een bekende manier om onderhoudbaarheid te meten is de zogenaamde Maintainability Index (MI). Hier is echter veel kritiek op [28]–[30].

2.3 Architectuur

Bass, L. en Clements, P. definiëren de architectuur van software met de volgende definitie [31]: "The architecture of a software-intensive system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them". Ook noemen Bass en Clements vier verschillende aspecten die behoren bij software-architectuur:

- **Statische structuur** (interne design-time elementen zoals modules, classes, packages, services, of andere zelfstandige code-eenheden en hun opstelling.)
- **Dynamische structuur** (de runtime-elementen zoals informatie-flows, parallelle of opeenvolgende uitvoering van interne taken, of de invloed die ze hebben op data en hun interacties)
- **Extern zichtbaar gedrag** (de functionele interacties tussen het systeem en zijn omgeving. Denk aan Informatie-flows in en uit het systeem, of API's.)
- **Kwaliteitseigenschappen** (externe zichtbare, niet-functionele eigenschappen van een systeem zoals prestaties, beveiliging of schaalbaarheid.)

IEEE 1471 A software-intensive system is any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole

IEEE 1471 The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. [32]

Albin, S. T. definieert software-architectuur als "De waarneembare eigenschappen van een softwaresysteem" [33]. Software-architectuur houdt zich bezig met het ontwerp en de implementatie van de structuur op hoog niveau [13].

Meerdere boeken nemen de definitie van [31] als uitgangspunt [34], [35]. Daarom zal in dit onderzoek ook gebruik gemaakt worden van deze definitie.

Bijlage A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

Literatuurlijst

[1] The Linux Foundation. (2019). Production-grade container orchestration, [Online]. Available: https://kubernetes.io.

[2] Docker Inc. (2019). Enterprise container platform for high-velocity innovation, [Online]. Available: https://docker.com.

[3] Red Hat inc. (2019). How ansible works, [Online]. Available: https://www.ansible.com/overview/how-ansible-works.

[4] Amber Ankerholz. (Apr. 2016). 8 container orchestration tools to know, [Online]. Available: https://www.linux.com/news/8-open-source-container-orchestration-tools-know/.

[5] A. M. Andreas Wittig, "Amazon web services in action", 2016. [Online]. Available: https://s3-ap-southeast-1.amazonaws.com/tv-prod/documents%2Fnull-Amazon+Web+Services+in+Action.pdf.

[6] Red hat inc. (2019). Ansible tower, [Online]. Available: https://www.ansible.com/products/tower.

[7] Chef. (2019). Chef, [Online]. Available: https://chef.io.

[8] Puppet. (2019). Unparalleled infrastructure automation and delivery, [Online]. Available: https://puppet.com.

[9] HashiCorp. (2019). Introduction to terraform, [Online]. Available: https://www.terraform.io/intro/index.html.

[10] T. G. Peter Mell, "The nist definition of cloud computing", p. 1, Oct. 2011. DOI: https://doi.org/10.6028/NIST.SP.800-145. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final.

[11] A. Wiggins, "The twelve-factor app", 2017. [Online]. Available: https://12factor.net.

[12] D. D. Gereld Weber, *Trends in Enterprise Application Architecture*. Feb. 2006, vol. 4437.

[13] P. Kruchten, "Architectural blueprints—the "4+1" view model of software architecture", pp. 42–50, Nov. 1995.

[14] D. A. Wheeler, "Why open source software/free software (oss/fs)? look at the numbers", Nov. 2004. [Online]. Available: http://www.robotcub.org/index.php/robotcub/content/download/290/1049/file/Why%20Open%20Source%20Software.pdf.

[15] K. de Munter, "Stageplan en oriëntatie developers.nl", 2017.

[16] Developers.nl, "Positioneringsprofiel developers.nl", 2018.

[17] M. A. Nieuwenhuis, *The Art of Management (the-art.nl)*. 2010, ISBN: 978-90-806665-1-1. [Online]. Available: http://123management.nl/0/030_cultuur/a300_cultuur_02_typering.html.

[18] K. de Munter, "Afstudeervoorstel", 2019.

[19] M. D. Hill, "What is scalability?", vol. 18, pp. 18–21, 4 Dec. 1990. [Online]. Available: https://dl.acm.org/citation.cfm?id=121975.

[20] T. W. Leticia Duboc David S. Rosenblum, "A framework for modelling and analysis of software systems scalability", May 2006. [Online]. Available: http://discovery.ucl.ac.uk/4990/1/4990.pdf.

[21] A. B. Bondi, "Characteristics of scalability and their impact on performance", Sep. 2000. [Online]. Available: https://www.win.tue.nl/~johanl/educ/2II45/2010/Lit/Scalability-bondi%202000.pdf.

[22] M. A.-E.-B. Hesham El-Rewini, *Advanced computer architecture and parallel processing*. 2005.

[23] J. B. G. Charles B. Weinstock, "On system scalability", 2006. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7887.

[24] A. A. T. Penny Grubb, *Software Maintenance: Concepts And Practice (Second Edition)*. River Edge, N.J.: World Scientific, 2003, vol. 2.

[25] IEEE, "Standard glossary of software engineering terminology", IEEE 610.12, 1990.

[26] J. K. C. Krishan K. Aggarwal Yogesh Singh, "An integrated measure of software maintainability", 2002. DOI: https://doi.org/10.1109/RAMS.2002.981648.

[27] ISO, "Software product quality", International Organization for Standardization, Geneva, CH, ISO 25010, 2011.

[28] O. T. Berna Seref, "Software code maintainability: A literature review", vol. 7, 3 May 2016. [Online]. Available: http://aircconline.com/ijsea/V7N3/7316ijsea05.pdf.

[29] R. Niedermayr, "Why we don't use the software maintainability index", Mar. 2016. [Online]. Available: https://www.cqse.eu/en/blog/maintainability-index.

[30] A. van Deursen, "Think twice before using the "maintainability index"", Aug. 2014. [Online]. Available: https://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/.

[31] P. C. Len Bass, *Software Architecture in Practice*. Pearson Education (US), Sep. 2012, ISBN: 9780321815736.

[32] IEEE, "Standard glossary of software engineering terminology", IEEE 1471, 2000.

[33] S. T. Albin, *The art of software architecture: design methods and techniques*. Wiley Publishing, Inc., Indianapolis, Indiana, Mar. 2003, vol. 9, ISBN: 9780471468295.

[34] E. W. Nick Rozanski, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Nov. 2011, ISBN: 9780132906074.

[35] R. K. Humberto Cervantes, *Designing Software Architectures: A Practical Approach*. Addison-Wesley Professional, 2016, ISBN: 9780134390789.