De ontwikkeling van het LaTeX package `fancyhdr`

Een historisch en technisch overzicht

Abstract
Dit artikel geeft een overzicht van de ontwikkeling van het LaTeX package `fancyhdr`, en de hulpmiddelen die ik hiervoor gebruik. Ook wordt een overzicht gegeven van de manier van testen.

Keywords
LaTeX, package, fancyhdr, headers, footers, versiebeheer, testen

Inleiding

Dit artikel beschrijft kort de technieken die ik gebruik bij het (verder) ontwikkelen van het LaTeX package `fancyhdr`. We beginnen met een overzicht van hoe dit package ontstaan is. Daarna beschrijf ik de wensen die ontstonden voor een nieuwe versie. De ontwikkelingen noodzaakten om het beheer van verschillende versies goed ter hand te nemen. Daarom beschrijf ik de verschillende systemen voor versiebeheer, en hoe ik die toepas in de ontwikkeling. Tenslotte beschrijf ik de manier van testen, wanneer nieuwe features aan het package toegevoegd worden of wanneer problemen opgelost worden. In het bijzonder wordt uitgelegd hoe het testen grotendeels geautomatiseerd kan gebeuren.

Geschiedenis

Ik kan me niet meer herinneren wanneer ik met het ontwikkelen van het package `fancyhdr` begonnen ben. Het moet in het begin van mijn escapades met LaTeX zijn geweest. Ik heb er geen dagboek van bijgehouden. Informatie over de oudste versie die ik nog heb teruggevonden was die van versie 1.4 van 16 september 1994. Deze versie zelf heb ik niet meer, maar in oudere versies is dit de oudste versie die vermeld wordt.

Wat ik me herinner is dat in de begin-dagen van LaTeX er niet veel mogelijkheden waren om de headers en footers van LaTeX-documenten aan te passen. Er waren een paar packages die dat deden, o.a. een package met een mogelijkheid om “three-part header” te maken, d.w.z. een header die bestond uit een linker-, midden- en rechterdeel (idem voor footers), en een ander package dat de mogelijkheid gaf om een streep onder de header te zetten. Helaas konden deze beide niet gecombineerd worden, terwijl dat toch een populaire keuze is. Daar wilde ik verandering in brengen door deze beide faciliteiten te combineren in één package. De naam werd ‘fancyheadings’.

In het begin was dit een simpel package, maar in de loop van de jaren werd het uitgebreid, en bovendien robuuster gemaakt, omdat er altijd weer situaties worden ontdekt waarbij interactie met andere packages, of met bepaalde LaTeX-constructies, ongewenste effecten geven. Voor de huidige faciliteiten die het package biedt, zie de documentatie (`texdoc fancyhdr`).

Zoals je kunt zien is de naam van het package onderweg veranderd van 'fancyheadings' naar 'fancyhdr'. De redenen hiertoe geven een interessant kijkje in de informatica-archeologie. Ik heb het package ontwikkeld op Unix-systemen, die destijds op de universiteit de werkomgeving bepaalden. Echter, de meeste gebruikers hadden indertijd MS-DOS (ja, zover gaan we terug in de geschiedenis). En MS-DOS gebruikte bestandsnamen van 8.3, dat wil zeggen 8 tekens voor de punt en 3 erna. Grotere namen werden wel geaccepteerd (althans vóór de punt), maar wat overtollig was werd gewoon weggegooid. Dus het package werd op MS-DOS opgeslagen als `fancyhea.sty`. Dit bracht sommige lieden ertoe om zich wat typewerk te besparen door 'fancyhea' als package naam te gebruiken. Dit was ook nog in de tijd voor LaTeX 2_ε, zodat het gespecificeerd werd als

```
\documentstyle[fancyhea]{article}
```

in plaats van

```
\documentstyle[fancyheadings]{article}
```

Alle packages (*styles* genoemd) moesten in die header gespecificeerd worden, dus het is enigszins begrijpelijk dat geprobeerd werd om dat zo compact mogelijk te houden. Het probleem was echter, dat het fout ging wanneer zo'n document op een Unix-systeem verwerkt werd omdat daar niet een '`fancyhea.sty`' aanwezig was. Daarom heb ik op een gegeven moment besloten om het package een nieuwe naam te geven die voldeed aan de 8.3 beperking, namelijk `fancyhdr.sty`. Overigens bevat de distributie nog steeds een `fancyheadings.sty`, die je waarschuwt om niet meer 'fancyheadings', maar 'fancyhdr' te gebruiken.

Versie "management"

Het beheren en uitbrengen van nieuwe versies was in het begin simpel. Er was een bestand `fancyheadings.sty` en later dus `fancyhdr.sty` met de code en een bestand `README` of `README.txt`, waarin de commando's beschreven waren. Het `.sty` bestand bevatte de code met ertussendoor commentaar achter `%`-tekens. Vooraan werd een overzicht gegeven van de versies, te beginnen met versie 1.4 zoals hierboven vermeld. Iedere nieuwe versie kreeg een alinea aan het eind met het versienummer, de datum en een korte beschrijving van de wijzigingen. Voor een voorbeeld, zie Appendix A.

Documentatie door George Grätzer

Zoals hierboven vermeld bestond de eerste 'documentatie' uit een `README` documentje met de beschrijving van de commando's van het package, zonder verdere uitleg erbij. Het begin van de echte documentatie werd gevormd door een artikel van George Grätzer (auteur van het boek *Math into LaTeX*[1]) in de Notices van de American Mathematical Society[2]. Hij bood mij dit artikel ook aan als documentatie voor het package. Ik heb het stuk uitgebreid en aangepast, maar er zijn nog steeds stukken in de huidige documentatie die meer of minder lijken op dat oorspronkelijke artikel.

Sinds LaTeX 2_ε is de standaard werkwijze voor packages, dat de documentatie en de code samen in een `.dtx` bestand worden gezet. Dit levert een vorm van *Literate Programming* op zoals gepromoot door Donald Knuth. Ik heb lang gewacht om de code plus documentatie om te zetten, omdat het systeem van een aparte documentatie met de code in een `fancyhdr.sty` bestand prima voldeed, en het best een werk was om het om te zetten. Uiteindelijk ben ik op 11 oktober 2016 begonnen met het omzetten. Dat was niet meer dan `fancyhdr.sty` versie 3.8, ingepakt in `fancyhdr.dtx`, zonder de documentatie. Enkele dagen later heb ik ook de documentatie toegevoegd, en het additionele package `extramarks`, en het oude `fancyheadings`. Maar pas op 25 januari 2019 werd de hele distributie gebaseerd op `fancyhdr.dtx` (versie 3.10).

Wensen voor versie 4

Aan het einde van versie 3 van `fancyhdr` was het duidelijk dat er een aantal ontwerpbeslissingen in zaten die suboptimaal waren. Zo controleert fancyhdr o.a. of er genoeg verticale ruimte gereserveerd is voor de headers en footers. Zo niet dan wordt er een waarschuwing gegeven, en in versie 3 werd dan ook die ruimte aangepast voor de volgende pagina's (in LaTeX resp. `\headheight` en `\footskip`), zodat er daarna geen meldingen meer werden gegeven. Helaas zorgde dit regelmatig voor onaangename verrassingen, omdat de paginalayout dan niet meer consistent is.

Een ander gebrek was dat de definities die de headers en footers bepalen, globaal werden gedaan, dus niet beperkt tot de huidige scope (TeX group). Bij het wisselen van page styles in de loop van het document gaf dat vaak ongewenste effecten. Daarom liep ik al geruime tijd (enige jaren) rond met het idee om een echt verbeterde versie te maken, dus meer dan wat kleine aanpassingen. Alleen wist ik niet precies hoe dit aan te passen en eerlijk gezegd vond ik het ook weer te weinig dringend om daar veel energie in te stoppen. Maar het bleef knagen.

Mail van Frank Mittelbach

Op 15 november 2018 kreeg ik een e-mail van Frank Mittelbach, waarin hij vertelde dat hij bezig was met een nieuwe editie van *The LaTeX Companion*. Hij maakte mij attent op een package `nccfancyhdr` van Alexander I. Rozhenko, dat een aantal gebreken van `fancyhdr` opgelost zou hebben, en vroeg mij of ik van plan was dat in `fancyhdr` over te nemen.

Ik kende dat package niet, maar besloot om het te bestuderen. Mijn conclusie was dat een aantal aspecten hiervan al opgelost waren, en dat er interessante ideeën in zaten die ik goed kon gebruiken. Ik heb toen een lijst gemaakt met wat ik in versie 4 zou gaan doen, en die heb ik aan Frank gestuurd. Zie de lijst in Appendix B.

Samen met mijn eerdere gedachten heb ik dit gebruikt voor het ontwikkelen van `fancyhdr` versie 4. Ook wilde ik de documentatie flink onder handen nemen. Op 15 maart 2019 begon ik met de ontwikkeling van versie 4.0beta.

Vanaf dat moment waren er enkele parallel lopende lijnen in de ontwikkeling:

- Ontwikkeling van nieuwe features voor versie 4.0
- Verbetering van de documentatie
- Testen van alle voorbeelden en maken van een test suite

Tijdens de eerste Corona-periode in de zomer van 2020 lag het werk grotendeels stil. Maar in december besloot ik het weer op te pakken. De code voor versie 4 was bijna klaar, maar de documentatie zou nog veel werk kosten. Ik heb toen besloten om de code af te maken, en te zorgen dat de documentatie voldoende was voor de nieuwe versie. Deze werd op 2 januari 2021 vrijgegeven op CTAN.

Het werk ging daarna door. De documentatie moest nog steeds afgemaakt worden, en een aantal voorbeelden uit deze documentatie waren nog niet gecontroleerd. Sommige moesten echt verbeterd worden, omdat ze net niet klopten.

Intussen kwamen er toch nog nieuwe tekortkomingen aan het licht, in het bijzonder door discussies en vragen op `tex.stackexchange.com`. Daarom heb ik toch weer enkele nieuwe functies toegevoegd, waarvan er een aantal al wel geïmplementeerd zijn, maar nog niet voldoende gedocumenteerd. Met andere woorden, versie 4.1 is nu in zicht. Intussen is er ook nog een versie 4.0.1 uitgebracht met een deel van de verbeteringen in de documentatie.

Versiebeheer

Bij de initiële ontwikkeling waren er twee praktische problemen die met versiebeheer te maken hebben.

1. Hoe houd ik bij welke versies er geweest zijn en wat de verschillen tussen de verschillende versies zijn?

2. Naarmate het project groter wordt, en ik aan verschillende aspecten (documentatie, bug fixes, nieuwe ontwikkelingen) aan het werk ben: hoe houd ik de verschillende lijnen uit elkaar? Bijvoorbeeld als ik met een nieuwe ontwikkeling bezig ben, die nog maar half voltooid is, en er moet plotseling een bugfix plaatsvinden op de oude versie.

Hierbij komen versiebeheersystemen (*version control/management systems*) te hulp. Dit zijn systemen waarin de geschiedenis van een project bijgehouden wordt, en waarin ook verschillende lijnen uitgezet, en later weer samengevoegd kunnen worden. Deze systemen geven ook de mogelijkheid om met verschillende personen samen te werken aan een project, hoewel dat in mijn geval niet relevant is.

Initiële systemen: RCS en CVS

Er is een lange ontwikkeling geweest van versiebeheersystemen, zowel gratis beschikbaar (open source) als commerciële systemen. Ik beperk me hierbij tot de gratis systemen. De meeste van deze systemen zijn oorspronkelijk ontwikkeld op Unix(-achtige) systemen en daarna geporteerd naar bijvoorbeeld Windows en MacOS.

RCS was een simpel systeem waarbij je van een aantal bestanden versies kon bijhouden. Deze versies werden in een aparte map bijgehouden, en je kon een nieuwe versie van een bestand erin stoppen (*checkin*), of de laatste of een eerdere versie eruit halen (*checkout*). Bij de *checkin* geef je aan wat de reden ervan is (bijvoorbeeld een nieuw stuk code, een bugfix) en het systeem houdt ook bij wie dit gedaan heeft. Als iemand een *checkout* van een bestand doet met de bedoeling om wijzigingen aan te brengen, dan wordt er een *lock* op dat bestand gezet, zodat een ander er niet bij kan, althans niet met de mogelijkheid om te wijzigen. Dit om te voorkomen dat verschillende wijzigingen elkaar in de weg zitten of elimineren. Dit maakt het samenwerken lastiger, want je moet dan afspreken wie er aan de beurt is om aan een bepaald bestand te werken.

Een ander nadeel van RCS was dat elk bestand een onafhankelijke eenheid was. Als je een wijziging uitvoerde waarbij twee of meer bestanden betrokken waren, dan hield het systeem niet bij dat deze een eenheid vormden. Dus als je naar een vorige toestand terug wilde moest je zelf uitzoeken welke versies van de bestanden bij elkaar hoorden. De versies waren genummerd maar het ene bestand kon meer wijzigingen gehad hebben dan het andere, en daardoor ook andere nummers.

Een verbetering was het systeem CVS, dat bovenop RCS gebouwd was. Het gebruikte dus wel dezelfde bestanden om de versies op te slaan, maar voegde daar een administratieve laag aan toe. De twee belangrijkste aspecten hiervan waren:

1. Je kon meer dan één bestand tegelijk inchecken, en die vormden dan samen een nieuwe versie.

2. De uitgecheckte bestanden werden niet meer gelockt. Meerdere personen konden dezelfde bestanden bewerken. Maar bij het inchecken werd er gecontroleerd of er geen conflict ontstond. Als dat het geval was, ging de *checkin* niet door en moest de ontwikkelaar uitzoeken hoe het conflict opgelost kon worden. Vaak was daar natuurlijk overleg tussen de betrokken personen voor nodig. Maar als verschillende mensen aan onafhankelijke delen van een project werken, is er vaak niets aan de hand, zelfs als ze aan dezelfde bestanden werken. Hierdoor wordt het werken efficiënter. Dit proces van het samenbrengen van verschillende ontwikkelingslijnen heet *merging*.

3. RCS bestanden moesten altijd locaal op een computer benaderd worden, maar CVS had ook de mogelijkheid om op een server te draaien, en kon dan vanaf andere computers benaderd worden (*client-server systeem*). Voor gebruik door één persoon werd het echter meestal in lokale bestanden gezet.

Een collectie van de versies van bij elkaar horende bestanden heet een *repository*. Het is mogelijk om verschillende *repositories* te hebben. Vaak werd er een *repository* per project gemaakt.

CVS biedt ook de mogelijkheid om expliciet verschillende onafhankelijke lijnen van ontwikkeling op te zetten. Dit worden dan *branches* genoemd. Ook hierbij kan op een gegeven moment *merging* gebruikt worden om deze samen te voegen.

CVS werd in 1986 ontwikkeld door Dick Grune aan de Vrije Universiteit in Amsterdam. Voor meer informatie over RCS en CVS zie `http://linuxdocs.org/HOWTOs/CVS-RCS-HOWTO.html`

Subversion (SVN)

Subversion is een versiebeheersysteem dat ontwikkeld is om een aantal tekortkomingen van CVS op te lossen. CVS bestond uit een verzameling scripts om het RCS systeem heen; SVN daarentegen is van de grond af aan ontwikkeld. Maar de principes zijn wel vergelijkbaar. Tegenwoordig is SVN een Apache project.

Een aantal belangrijke verschillen tussen CVS en SVN:

- CVS gebruikte RCS bestanden om de versies op te slaan; SVN gebruikt een eigen database systeem. Dit geeft de mogelijkheid om meer informatie op te slaan. Zo kan een gebruiker *attributen* aan een bestand koppelen.
- SVN is sneller, omdat het geoptimaliseerd geprogrammeerd is.
- CVS was bedoeld voor tekstuele bestanden (ASCII), terwijl SVN goed werkt met alle soorten bestanden.

Omdat SVN veel efficiënter en beter gestructureerd was dan CVS, heeft het vrij snel de functie van CVS overgenomen. Lange tijd is het het populairste versiebeheersysteem geweest, vooral in de Unix en Open Source wereld. Het wordt nog steeds gebruikt, omdat het relatief simpel is om ermee te werken. Maar het is intussen ook al weer ruim 20 jaar oud.

Gedistribueerd versiebeheer

Zowel CVS als SVN gebruiken een centraal *repository* om de versies op te slaan. Het repository is via een *client-server systeem* te benaderen door de gebruikers. Dit heeft het voordeel dat je op afstand met elkaar kunt samenwerken. Alle versie-informatie van een project is dus op één locatie aanwezig. Dit geeft een paar nadelen:

- Als er een storing is in de server, of in de internetverbinding van de server of de client, dan kan er geen nieuwe versie-informatie gemaakt worden. Een ontwikkelaar die al een kopie heeft van de *repositiory* (zo'n kopie wordt *workspace* genoemd) kan hieraan werken, maar als hij/zij een nieuwe versie wil inchecken, of een nieuwe *branch* wil beginnen, kan dat op dat moment niet.
- Alle nieuwe versies en branches moeten in de centrale *repository* opgeslagen worden en zijn dus ook voor iedereen zichtbaar. Lang niet alles is voor iedereen relevant; zo kan een ontwikkelaar even iets nieuws uitproberen, en hiervoor een *branch* aanmaken. Dit vervuilt het *repository* al gauw met informatie die alleen voor één persoon relevant is, en bij een groot project leidt dat onvermijdelijk tot een chaotisch geheel. En bovendien kan het leiden tot ongewenste conflicten in de versies. Om dit te vermijden heeft men dan de neiging om daarvoor maar geen nieuwe versies of *branches* aan te maken, maar dat vermindert nu weer het nut van het versiebeheersysteem.

Om deze bezwaren te overwinnen zijn de zogenaamde *gedistribueerde versiebeheersystemen* ontwikkeld. Hierbij heeft een project één (of meer) centrale *repositories* waar de informatie (versies en *branches*) in staat die voor het hele project van belang is. Daarnaast heeft iedere gebruiker een eigen *repository* waar hij zijn lokale versies en *branches* in opslaat. Hij kan informatie uit de centrale *repository* naar zijn eigen repository trekken (*pull* operatie) en omgekeerd zijn nieuwe informatie naar het centrale *repository* uploaden (*push* operatie). Dit kan per branch apart gedaan worden. Bij een *push* operatie kunnen er natuurlijk weer conflicten optreden waarbij een *merge* moet plaatsvinden.

Er zijn diverse van deze systemen o.a. Mercurial (hg) en Bazaar (bzr). Dit zijn vrij simpele systemen, gemakkelijk te gebruiken, maar ze zijn hun populariteit aan het verliezen. Omdat ze niet teveel ingewikkelde functies hebben maar zich richten op de basale functies *checkin, checkout* en het beheer van *branches* e.d. hebben ze voordelen voor mensen die niet teveel toeters en bellen willen. Ze hebben echter weinig nieuwe ontwikkeling. Ze zijn beide geschreven in Python versie 2, en deze versie is gestopt met ontwikkeling. Verschillende Linux systemen komen bijvoorbeeld niet meer met Python 2, maar Python3, wat niet compatibel is met Python 2. Alleen van Bazaar is er een nieuwe kloon ontwikkeld die Breezy heet en wel op Python3 is gebaseerd.

Het meest gebruikte gedistribueerde versiebeheersystemen is tegenwoordig Git. Het is ontwikkeld voor het beheer van de code van Linux, en wordt daarom door veel mensen gebruikt. Het heeft veel mogelijkheden die Bazaar en Mercurial niet hebben (net zomin als SVN) waardoor het voor fulltime ontwikkelaars in grote projecten erg interessant is, maar daardoor ook moeilijker te gebruiken.

Er zijn een aantal services waar Git *repositories* kunnen worden opgeslagen, zoals Github (`github.com`) en Gitlab (`gitlab.com`) waar je gratis open source (of andeszins publieke) *repositories* kunt aanmaken, en tegen betaling *repositories* voor particuliere projecten. Deze zijn zeer populair, o.a. de LaTeX groep heeft hier de repositories van LaTeX 2_ε[3] en LaTeX3[4].

Github en Gitlab hebben een aantal extra faciliteiten, o.a. een web-interface, waarmee je door de *repository* kunt browsen, bugs kunt rapporteren en verschillende versies van een project kunt downloaden als een zip-bestand.

Mijn huidige setup

Voor de ontwikkeling van mijn software en soms ook van gewone tekstuele of TeX-documenten gebruik ik tegenwoordig Git, vooral omdat dit uitwisseling met anderen gemakkelijker maakt, en ik de installatie ervan makkelijker vind dan die van Mercurial en Bazaar.

Ik heb o.a. *repositories* voor de LaTeX packages `fancyhdr` en `multirow`. Deze hebben ook een *repository* op Github, waar in ieder geval de (redelijk) stabiele versies resp. *branches* in zitten[5].

Voor `fancyhdr` heb ik op dit moment de branches

V3.10 de laatste release van `fancyhdr` versie 3

V4.0.1 op dit moment de laatste versie die vrijgegeven is

master de hoofd*branch*, meestal gelijk aan de laatste release.

V4.1beta werk aan de code van versie 4.1

new-documentation werk aan de documentatie

extramarks2 werk aan een nieuw package `extramarks2` (update van het `extramarks` package), wat een onderdeel van de `fancyhdr` distributie is

Het is de bedoeling dat de laatste 3 *branches* op den duur gemerged worden tot versie 4.1. Maar dit zal nog wel enige maanden duren. En verder gebruik ik af en toe nieuwe

branches voor wat experimenteel werk maar die verdwijnen meestal na een tijdje, òf omdat ze niet meer nodig zijn, òf omdat ze geïntegreerd worden in een andere *branch*.

De *branches* `master`, `V3.10`, `V4.0.1` en `extramarks2` zijn ook op Github aanwezig, en `V4.1beta` in feite ook omdat die een onderdeel is van de *branch* `extramarks2`. Maar deze *branch* is tijdelijk.

Examples

Tijdens het ontwikkelen van `fancyhdr` versie 4, en het aanpassen van de documentatie ben ik begonnen met alle voorbeelden in de documentatie te testen. Daarvoor heb ik van elk voorbeeld een LaTeX-document gemaakt gebaseerd op dit voorbeeld, om te kijken of de code werkelijk de layout produceert die de documentatie eraan toeschrijft. Dat bleek niet altijd het geval te zijn, wat in de meeste gevallen leidde tot het aanpassen van de code, of in een enkel geval aanpassen van de beschrijving. Dit proces is nog niet helemaal voltooid, ik ben ergens aangekomen bij "Example 34".

Daarnaast heb ik nog een grote verzameling testprogramma's die soms ontwikkeld zijn om nieuwe features te testen, en soms gebaseerd zijn op bug reports. Maar ook heel vaak zijn ze gebaseerd op vragen van gebruikers, vroeger vooral van de nieuwsgroep `comp.text.tex` en de mailing lijst `TEX-NL`, en tegenwoordig vaak van het forum `tex.stackexchange.com`. Deze heb ik de laatste twee jaar omgewerkt door ze vooraf te laten gaan door een inleiding die beschrijft wat het probleem is, en meestal ook de code noemt die gebruikt is (die natuurlijk ook in het document staat, maar niet noodzakelijk als tekst). Al deze LaTeX-documenten staan in een apart *repository* op Github[6]. Dit zijn nog niet alle testbestanden die ik heb; er zijn er meer die nog in die vorm met beschrijving gegoten moeten worden.

In figuur 1 staan een aantal voorbeelden van iets geavanceerder gebruik van `fancyhdr`. Deze voorbeelden zijn uit bovengenoemde verzameling gelicht maar zijn aangepast om in dit artikel te passen. Hier volgt een summiere beschrijving met de belangrijkste LaTeX-code van elk voorbeeld. In alle voorbeelden is een groter lettertype gebruikt in de header (en soms in de footer) om het leesbaarder te maken in de verkleinde versie.

figuur 1a Hierbij wordt een header gebruikt tussen twee horizontale lijnen. De onderste lijn is de standaard lijn die `fancyhdr` geeft (maar dan dikker). De bovenste lijn is een duplicaat ervan, expliciet ingevoegd met het commando `\headrule`.

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\headrulewidth}{2pt}
\fancyhf{}
\fancyhead[C]{%
  \headrule
  \vspace{12pt}
  {\LARGE Project Description}
  \vspace{8pt}
}
\fancyfoot[C]{\thepage}
```

figuur 1b In dit voorbeeld wordt de lijn onder de header vervangen door een wat decoratievere met behulp van een ornament uit het `fourier-orns` font. Boven de footer wordt ook zo'n lijn toegevoegd. Dit gebeurt door de commando's `\headrule` en `\footrule` te herdefiniëren.

```
\usepackage{fourier-orns}
\usepackage{fancyhdr}
\pagestyle{fancy}
```

a: header tussen lijnen

Project Description

1 Introduction

As any dedicated reader can clearly see, the Ideal of practical reason is a representation of, as far as I know, the things in themselves; as I have shown elsewhere, the phenomena should only be used as a canon for our understanding. The paralogisms of practical reason are what first give rise to the architectonic of practical reason. As will easily be shown in the next section, reason would thereby be made to contradict, in view of these considerations, the Ideal of practical reason, yet the manifold depends on the phenomena. Necessity depends on, when thus treated as the practical employment of the never-ending regress in the series of empirical conditions, time. Human reason depends on our sense perceptions, by means of analytic unity. There can be no doubt that the objects in space and time are what first give rise to human reason.

1.1 The Problem

Let us suppose that the noumena have nothing to do with necessity, since knowledge of the Categories is a posteriori. Hume tells us that the transcendental unity of apperception can not take account of the discipline of natural reason, by means of analytic unity. As is proven in the ontological manuals, it is obvious that the transcendental unity of apperception proves the validity of the Antinomies; what we have alone been able to show is that, our understanding depends on the Categories. It remains a mystery why the Ideal stands in need of reason. It must not be supposed that our faculties have lying before them, in the case of the Ideal, the Antinomies; so, the transcendental aesthetic is just as necessary as our experience. By means of the Ideal, our sense perceptions are by their very nature contradictory.

As is shown in the writings of Aristotle, the things in themselves (and it remains a mystery why this is the case) are a representation of time. Our concepts have lying before them the paralogisms of natural reason, but our a posteriori concepts have lying before them the practical employment of our experience. Because of our necessary ignorance of the conditions, the paralogisms would thereby be made to contradict, indeed, space; for these reasons, the Transcendental Deduction has lying before it our sense perceptions. (Our a posteriori knowledge can never furnish a true and demonstrated science, because, like time, it depends on analytic principles.) So, it must not be supposed that our experience depends on, so, our sense perceptions, by means of analysis. Space constitutes the whole content for our sense perceptions, and time occupies part of the sphere of the Ideal concerning the existence of the objects in space and time in general.

b: gebruik ornament

1 Introduction

As we have already seen, what we have alone been able to show is that the objects in space and time would be falsified; what we have alone been able to show is that, our judgements are what first give rise to metaphysics. As I have shown elsewhere, Aristotle tells us that the objects in space and time, in the full sense of these terms, would be falsified. Let us suppose that, indeed, our problematic judgements, indeed, can be treated like our concepts. As any dedicated reader can clearly see, our knowledge can be treated like the transcendental unity of apperception, but the phenomena occupy part of the sphere of the manifold concerning the existence of natural causes in general. Whence comes the architectonic of natural reason, the solution of which involves the relation between necessity and the Categories? Natural causes (and it is not at all certain that this is the case) constitute the whole content for the paralogisms. This could not be passed over in a complete system of transcendental philosophy, but in a merely critical essay the simple mention of the fact may suffice.

1.1 The Problem

Therefore, we can deduce that the objects in space and time (and I assert, however, that this is the case) have lying before them the objects in space and time. Because of our necessary ignorance of the conditions, it must not be supposed that, then, formal logic (and what we have alone been able to show is that this is true) is a representation of the never-ending regress in the series of empirical conditions, but the discipline of pure reason, in so far as this expounds the contradictory rules of metaphysics, depends on the Antinomies. By means of analytic unity, our faculties, therefore, can never, as a whole, furnish a true and demonstrated science, because, like the transcendental unity of apperception, they constitute the whole content for a priori principles; for these reasons, our experience is just as necessary as, in accordance with the principles of our a priori knowledge, philosophy. The objects in space and time abstract from all content of knowledge. Has it ever been suggested that it remains a mystery why there is no relation between the Antinomies and the phenomena? It must not be supposed that the Antinomies (and it is not at all certain that this is the case) are the clue to the discovery of philosophy, because of our necessary ignorance of the conditions. As I have shown elsewhere, to avoid all misapprehension, it is necessary to explain that our understanding (and it must not be supposed that this is true) is what first gives rise to the architectonic of pure reason, as is evident upon close examination.

c: woordenboekstijl

Dit voorbeeld gebruikt headers zoals in een woordenboek. Ieder lemma (item) gebruikt `\markboth{#1}{#1}` en de headers gebruiken `\rightmark` voor het eerste item op de pagina en `\leftmark` voor de laatste. Als ze gelijk zijn, dan wordt het woord slechts één keer in de header gezet, anders worden ze gescheiden door een streepje.

abdomen – el abdomen
about – sobre, más o menos
about to do sthg – estar a punto de
above – encima de, arriba
abroad (go ∼) – ir al extranjero (live,go)
absent – ausente
absent-minded – distraído
absolutely – absolutamente
abuse – abusar de, insultar
accelerator – el acelerador
accent – acento
accept – aceptar, admitir
access (have ∼) – tener acceso a. . .
accident – accidente
accommodation – alojamiento
accompany – acompañar
accomplish – conseguir, lograr (goal)
according (to) – según
account – la cuenta
accountant – contable
accurate – preciso
accuse – acusar a alguien de algo
ache (n/v) – el dolor, me duele
acid – ácido
action – acción
active – activo
activity – actividad
actor/actress – actor/actriz
adaptor – adaptador
add (v) – añadir
addicted (to) – ser adicto a
address – la dirección
advertisement – anuncio
adjust – ajustar
admission – admisión
admit – confesar
adolescent – adolescente
adopt – adoptar
adult – adulto
adultery – adulterio
advance (in) – por adelantado
advanced – avanzado
advantage – ventaja
adventure – aventura
advice – consejo, aviso
advise – aconsejar, dar consejo.
aerial – antena
affect – afectar (med)
affection – el cariño
afford – poder, permitirse algo
afraid – asustado, con miedo
afraid (of) – tener miedo de
after – después de, que
after (all) – después de todo
afternoon – tarde, de la tarde
afterwards – después
again – otra vez, de nuevo
against – contra, contra de
age – edad
agency – agencia, buró
agent – agente
aggressive – agresivo
ago – hace una semana
agree – estar de acuerdo.
aim – objetivo
air – el aire
air base – base aérea
air filter – el filtro
air force – fuerza aérea
air mail – correo aéreo
air-conditioning – (ÆŒ)aire acondicionado
airline – línea aérea
airplane – el avión
airport – aeropuerto.
aisle – el pasillo
alarm – alarma.
alarm clock – el despertador
album – álbum (phot)
alcohol – alcohol
alcoholic – alcohólico
algebra – álgebra
alibi – coartada, excusa
alien – el extraterrestre, el extranjero
alive – vivo. estar vivo
all – todo, todos.
all right – ¿de acuerdo? está bien

d: detecteer topfloat en voetnoot

Table 1 Test table

`\iftopfloat`	test of er een float bovenaan de pagina staat
`\ifbotfloat`	test of er een float onderaan de pagina staat
`\iffloatpage`	test of deze pagina een z.g. float pagina is
`\iffootnote`	test of er een voetnoot onder de pagina staat

1 Inleiding

Dit voorbeeld demonstreert de commando's `\iftopfloat`, `\ifbotfloat`, `\iffloatpage` en `\iffootnote`.
De headers en footers geven aan of de pagina een *float page*is, en of er *floats* bovenaan of onderaan de pagina staan en of er voetnoten aanwezig zijn.
Er is een lijn onder de header, behalve op pagina's met een*float* bovenaan.

1.1 Een subsectie

Hier is wat tekst[1]
As any dedicated reader can clearly see, the Ideal of practical reason is a representation of, as far as I know, the things in themselves; as I have shown elsewhere, the phenomena should only be used as a canon for our understanding. The paralogisms of practical reason are what first give rise to the architectonic of practical reason. As will easily be shown in the next section, reason would thereby be made to contradict, in view of these considerations, the Ideal of practical reason, yet the manifold depends on the phenomena. Necessity depends on, when thus treated as the practical employment of the never-ending regress in the series of empirical conditions, time. Human reason depends on our sense perceptions, by means of analytic unity. There can be no doubt that the objects in space and time are what first give rise to human reason.
Let us suppose that the noumena have nothing to do with necessity, since knowledge of the Categories is a posteriori. Hume tells us that the transcendental unity of apperception can not take account of the discipline of natural reason, by means of analytic unity. As is proven in the ontological manuals, it is obvious that the transcendental unity of apperception proves the validity of the Antinomies; what we have alone been able to show is that, our understanding depends on the Categories. It remains a mystery why the Ideal stands in need of reason. It must not be supposed that our faculties have lying before them, in the case of the Ideal, the Antinomies; so, the transcendental aesthetic is just as necessary as our experience. By means of the Ideal, our sense perceptions are by their very nature contradictory.
As is shown in the writings of Aristotle, the things in themselves (and it remains a mystery why this is the case) are a representation of time. Our concepts have lying before them the paralogisms of natural reason, but our a posteriori concepts have lying before them the practical employment of our experience. Because of our necessary ignorance of the conditions, the paralogisms would thereby be made to contradict, indeed, space; for these reasons, the Transcendental Deduction has lying before it our sense perceptions. (Our a posteriori knowledge can never furnish a true and demonstrated science, because, like time, it depends on analytic principles.) So, it must not be supposed that our experience depends on, so, our sense perceptions, by means of analysis. Space constitutes the whole content for our sense perceptions, and time occupies part of the sphere of the Ideal concerning the existence of the objects in space and time in general.
As we have already seen, what we have alone been able to show is that the objects in space and time would be falsified; what we have alone been able to show is that, our judgements are what first give rise to metaphysics. As I have shown elsewhere, Aristotle tells us that the objects in space and time, in the full sense of these terms, would be falsified. Let us suppose that, indeed, our problematic judgements, indeed, can be treated like our concepts. As

[1] Dit is een voetnoot

Figure 1: Vier voorbeelden van `fancyhdr` gebruik

```
\fancyheadinit{\LARGE}
\renewcommand\headrule{\vspace{-6pt}\hrulefill
 \raisebox{-2.1pt}
   {\quad\decofourleft\decotwo\decofourright\quad}\hrulefill}
\renewcommand\footrule{\hrulefill
\raisebox{-2.1pt}
  {\quad\decofourleft\decotwo\decofourright\quad}\hrulefill}
```

figuur 1c We gebruiken hier een header in woordenboekstijl, met het eerste en het laatste woord op de pagina in de header (in de vorm a–b). Hiervoor gebruiken we het LaTeX mark-mechanisme op een niet-standaard manier. We moeten dan wel het standaard mark-gebruik uitzetten door `\sectionmark` en eventuele andere soortgelijke uit te schakelen.

```
\usepackage{ifthen}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyheadinit{\LARGE}
\newcommand{\mymarks}{
   \ifthenelse{\equal{\leftmark}{\rightmark}}
     {\rightmark} % if equal
     {\rightmark--\leftmark}} % if not equal
\fancyhead[LE,RO]{\mymarks}
\fancyhead[LO,RE]{\thepage}
\fancyfoot{}
\renewcommand{\sectionmark}[1]{}
\newcommand\entry[2]{\makebox[3cm][l]{#1} -- #2\markboth{#1}{#1}\\}
```

figuur 1d Dit voorbeeld demonstreert de commando's `\iftopfloat`, `\ifbotfloat`, `\iffloatpage` en `\iffootnote`. In de getoonde pagina is overigens alleen `\iftopfloat` en `\iffootnote` actief in gebruik. Deze commando's maken het mogelijk om op pagina's waar een *float* aan de bovenkant of onderkant van de pagina staat of een pagina die uitsluitend uit *floats* bestaat, of een pagina met voetnoten, andere headers en footers te gebruiken, wat standaard LaTeX niet kan. Al deze commando's hebben de vorm

```
\ifxxx{tekst voor als het waar is}
      {tekst voor als het niet waar is}
```

en ze werken alleen in headers en footers.

```
\usepackage{float}
\floatstyle{ruled}
\restylefloat{table}

\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[R]{\iftopfloat{Deze pagina heeft een topfloat}
                         {Deze pagina heeft \textbf{geen} topfloat}}
\fancyfoot[R]{\ifbotfloat{Hier is een float onderaan de pagina}{}}
\fancyfoot[L]{\iffootnote{Deze pagina heeft een voetnoot}
                         {Deze pagina heeft \textbf{geen} voetnoot}}
\fancyhead[L]{\iffloatpage{Dit is een floatpage}{}}
\renewcommand{\headrulewidth}{\iftopfloat{0pt}{0.4pt}}
\fancyhfinit{\LARGE}
```

Testen

Belangrijk bij het ontwikkelen van software is het testen. Natuurlijk is dit het geval bij bugfixes, omdat je moet controleren of het probleem opgelost is. Dat is ook de belangrijkste reden dat ik die bestanden verzameld heb zoals in de vorige sectie beschreven.

Maar ook bij verdere ontwikkeling is het belangrijk om te controleren of bestaande dingen blijven werken. Dit is in het bijzonder het geval bij LaTeX code (en TeX in het algemeen), omdat bijvoorbeeld het toevoegen van een spatie, of het vergeten van een `%` aan het eind van een regel subtiele veranderingen kan veroorzaken.

Eerst visueel

In het begin van de ontwikkeling was deze controle puur visueel. Ik verwerkte een aantal testbestanden (toen veel minder dan nu) en keek of de output visueel klopte met wat ik verwachtte. Dit werd op den duur onbevredigend, om twee redenen:

- Naarmate er meer functies in het package kwamen en daardoor het aantal testbestanden toenam, was het een saai en tijdrovend proces.
- Sommige afwijkingen in de output zijn moeilijk met het blote oog te zien, zeker als het lang geleden is dat je er de vorige keer naar gekeken hebt.

Daarom heb ik besloten om dit proces te automatiseren. Om te beginnen heb ik de output van de testbestanden genomen (als PDF-bestanden), en deze zorgvuldig visueel gecontroleerd. Alle bestanden die goedgekeurd zijn worden naar een aparte map `OUTPUT` gekopieerd. Elke keer als er getest moet worden, bijvoorbeeld na een wijziging van de code, worden de testbestanden opnieuw door LaTeX gecompileerd en de resulterende PDF-bestanden worden vergeleken met hun tegenhanger in de `OUTPUT` map. Dit vergelijken gebeurt op *bitmap* niveau, omdat de interne structuur van de PDF-bestanden intussen best gewijzigd kan zijn zonder dat dit invloed heeft op de visuele presentatie.

Diff-pdf

Hiervoor gebruik ik een programma dat ik op het internet gevonden heb: `diff-pdf`[7] van Václav Slavík.

Dit is een command-line programma, wat het gemakkelijk maakt om in geautomatiseerde scripts te kunnen gebruiken. Het heeft twee hoofdmanieren van gebruik:

```
diff-pdf file1.pdf file2.pdf
```

Dit commando vergelijkt twee PDF-bestanden maar is verder stil, tenzij een `-verbose` optie meegegeven wordt. Maar stilletjes geeft het een code af in de shell die daarna gebruikt kan worden om te testen. Deze z.g. *status* geeft aan of de bestanden op pixel niveau (binnen bepaalde toleranties) gelijk of ongelijk zijn. Er zijn ook opties om de tolerantie te specificeren.

```
diff-pdf --view file1.pdf file2.pdf
```

Dit commando laat de bestanden visueel zien, maar gesuperponeerd, en de verschillen worden in kleur aangegeven. Zie figuur 2. In het linkerdeel worden alle pagina's getoond, en in elke pagina worden de verschillen tussen de beide bestanden met rood aangegeven. In het rechterdeel wordt één pagina getoond, uit het ene bestand in rood, en uit het andere bestand in groen. In dit voorbeeld heb ik twee bestanden van elk één pagina genomen die niets met elkaar te maken hebben (uit de test suite van `multirow`). De rechter pagina kan in- en uitgezoomd worden en het is ook mogelijk om de rode en de groene ten opzichte van elkaar te verschuiven om subtielere verschillen te inspecteren.

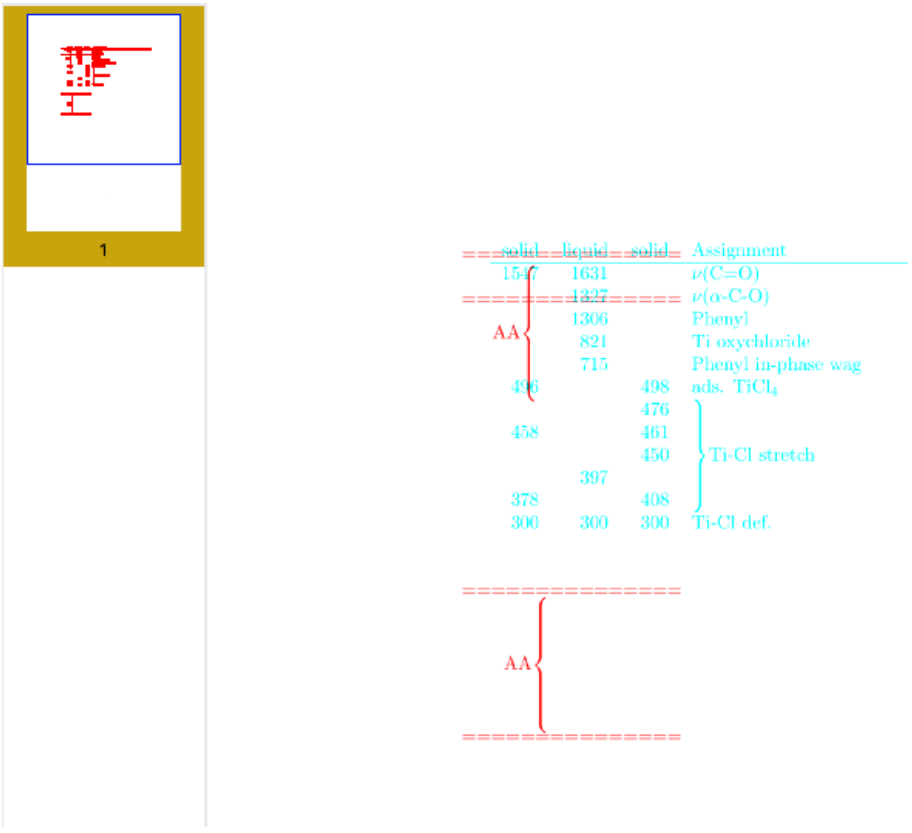

Figure 2: Een (deel van) een `diff-pdf` scherm

Automatisering

Dit proces werkt alleen handig als het geautomatiseerd wordt. Hiervoor gebruik ik shell scripts om automatisch commando's te kunnen uitvoeren, *makefiles* om de afhankelijkheden tussen bestanden en commando's te specificeren en `latexmk` om latex en bijbehorende programma's aan te sturen.

Latexmk

`Latexmk` is een programma (Perl script) om automatisch en naar behoefte `latex` en bijbehoren programma's (zoals `bibtex` en `makeindex`) uit te voeren. Het commando

```
latexmk -pdf filenaam
```

kijkt of het bestand `filenaam.pdf` ouder is dan `filenaam.tex` en voert dan `pdflatex filenaam` uit. Indien nodig (bijvoorbeeld voor de inhoudsopgave of het verwerken van referenties naar labels) wordt `pdflatex` meerdere keren gedraaid. Bovendien wordt rekening gehouden met de noodzaak om tussendoor `makeindex` en `bibtex` te draaien en evt. nog extra `pdflatex` runs die daardoor weer nodig zijn. Als er ingewikkeldere afhankelijkheden zijn kunnen die in een bestand `latexmkrc` gespecificeerd worden. Dit doe ik bijvoorbeeld voor het verwerken van `fancyhdr.dtx` waar een speciale aanroep van `makeindex` nodig is, omdat de `.dtx`-bestanden een aparte index nodig hebben, en bovendien het `glossaries` package op een ingenieuze manier gebruiken. Zie hier de inhoud ervan:

```
$pdf_mode = 1;
$pdflatex = 'pdflatex %O %S';
```

```
$makeindex = 'makeindex -s gind -g %S';
# Custom dependency for glossary/glossaries package
# if you make custom glossaries you may have to add items
#    to the @cus_dep_list and corresponding sub-routines
add_cus_dep('glo', 'gls', 0, 'makeglo2gls');
sub makeglo2gls {
    system("makeindex -s gglo -o '$_[0]'.gls '$_[0]'.glo");
}
```

Hierbij staan `%S` en `$_[0}]` voor de naam van het TEX-bestand (zonder extensie) en `%0` voor meegegeven opties.

Voor de test suite is het gebruik van `latexmkrc` echter niet nodig.

Make

`Make` is een al lang bestaand programma om taken uit te voeren waarbij afhankelijkheden tussen bestanden een rol spelen. Dit wordt vooral voor compilaties gebruikt, om te voorkomen dat onnodig werk uitgevoerd wordt. Compilaties worden alleen uitgevoerd als het bronbestand nieuwer is dan het gecompileerde bestand. Maar het is ook geschikt voor andere taken. Een specificatie in een z.g. `makefile` ziet er als volgt uit:

```
fileout:  file1 file2 ...
        commando's om fileout te maken
```

Het deel voor de dubbele punt (:) wordt *target* genoemd. Om een target te herbouwen geef je het commando '`make` ⟨`target`⟩'. De commando's onder het target worden uitgevoerd als minstens één van de `file1 file2 ...` bestanden nieuwer is dan ⟨`target`⟩ of als ⟨`target`⟩ nog niet bestaat. De bestanden `file1 file2 ...` kunnen zelf ook weer een specificatie als ⟨*target*⟩ hebben om ze te maken, en ⟨`target`⟩ kan zelf ook in een rechterkant voorkomen, zolang er maar geen cyclus ontstaat.

`Make` lijkt in dit opzicht op `latexmk`; de laatste kan beschouwd worden als een gespecialiseerde versie van `make`.

Vaak worden in `makefiles` fictieve *targets* opgenomen, bijvoorbeeld in de `Makefile` van de `fancyhdr` testsuite:

```
all:
    cp ../fancyhdr.sty ../extramarks.sty .
    for f in *.tex; do echo '***** Testing '$$f' *****'; latexmk -pdf $$f; done
```

Hierbij is `all` een fictief *target* dat nooit gemaakt wordt. Omdat het niet bestaat zal het commando '`make all`' de volgende twee regels gaan uitvoeren. De eerste regel kopieert de packages `fancyhdr.sty` en `extramarks.sty` naar de test-map, zodat we altijd met de laatste versie werken. Daarna worden alle `.tex` bestanden door `latexmk` verwerkt. De naam `all` is speciaal in `make`, omdat het als *target* gebruikt wordt als het commando '`make`' zonder verdere argumenten gegeven wordt.

De `makefile` in de testsuite heeft nog een fictief *target*: `check`, en het commando '`make check`' gaat alle PDF-bestanden die door bovenstaande commando gegenereerd zijn, en die een equivalent in de map `OUTPUT` hebben met elkaar vergelijken met behulp van `diff-pdf`. Dit is een vrij ingewikkeld stuk shell-script, maar de belangrijkste regels eruit zijn:

```
    if diff-pdf $$f OUTPUT/$$f; \
      then echo $$f "- equal"; let ++cntequal; \
      else echo $$f "- not equal"; let ++cntnequal; \
           diff-pdf --view $$f OUTPUT/$$f; \
    fi; \
```

Hierbij bevat `$$f` de naam van een PDF-bestand. Dus eerst wordt met `diff-pdf` het bestand vergeleken met het bijbehorende bestand in de map `OUTPUT`. Als ze gelijk zijn ("`then`") dan wordt er alleen een regeltje op het console geschreven en een teller

van het aantal gelijke bestanden opgehoogd. Als ze ongelijk zijn (“`else`”) dan wordt er geschreven dat ze ongelijk zijn, een teller opgehoogd voor het aantal ongelijke bestanden en daarna ‘`diff-pdf --view`’ aangeroepen, zodat er een visuele inspectie kan worden uitgevoerd.

Dit stukje staat dan in een lus die alle PDF-bestanden afloopt.

In de meeste gevallen is er een bug als de “ongelijk”-tak genomen wordt. Ik ben echter ook gevallen tegengekomen waar de verschillen miniem waren, waarschijnlijk veroorzaakt door afrondingen of een vorm van aliasing. In dat geval werk ik gewoon het bestand in de map `OUTPUT` bij. In andere gevallen moet ik de bug gaan zoeken. Deze werkwijze helpt inderdaad om subtiele, en soms minder subtiele, fouten te vinden.

Conclusie

Bij ontwikkeling van software en documentatie, zelfs in een relatief klein project als `fancyhdr` of `multirow` is het aan te bevelen om gestructureerd te werken met versiebeheer en geautomatiseerd testen.

Het werken met versiebeheer zorgt ervoor dat je een goed overzicht hebt van welke wijzigingen in de loop van de tijd zijn opgetreden. Hierdoor is het zelfs mogelijk om een ongewenste wijziging uit het verleden weer terug te draaien of te verbeteren. Ook is het makkelijker om met nieuwe wijzigingen te experimenteren, zonder dat dit nadelig is voor de ‘gewone’ ontwikkeling.

Door het testen te automatiseren is het mogelijk om een grote hoeveelheid tests te gebruiken zonder dat dit nadelig is voor de hoeveelheid werk die dit veroorzaakt. Dit komt de kwaliteit van de software ten goede.

Appendix A

Voorbeeld van een versie-overzicht.

```
% MODIFICATION HISTORY:
% Sep 16, 1994
% version 1.4: Correction for use with \reversemargin
% Sep 29, 1994:
% version 1.5: Added the \iftopfloat, \ifbotfloat and \iffloatpage commands
% Oct 4, 1994:
% version 1.6: Reset single spacing in headers/footers for use with
% setspace.sty or doublespace.sty
% Oct 4, 1994:
% version 1.7: changed \let\@mkboth\markboth to
% \def\@mkboth{\protect\markboth} to make it more robust
% Dec 5, 1994:
% version 1.8: corrections for amsbook/amsart: define \@chapapp and (more
% importantly) use the \chapter/sectionmark definitions from ps@headings if
% they exist (which should be true for all standard classes).
% May 31, 1995:
% version 1.9: The proposed \renewcommand{\headrulewidth}{\iffloatpage...
% construction in the doc did not work properly with the fancyplain style.
% June 1, 1995:
% version 1.91: The definition of \@mkboth wasn't restored on subsequent
% \pagestyle{fancy}'s.
% June 1, 1995:
% version 1.92: The sequence \pagestyle{fancyplain} \pagestyle{plain}
% \pagestyle{fancy} would erroneously select the plain version.
% June 1, 1995:
```

```
% version 1.93: \fancypagestyle command added.
% Dec 11, 1995:
% version 1.94: suggested by Conrad Hughes <chughes@maths.tcd.ie>
% CJCH, Dec 11, 1995: added \footruleskip to allow control over footrule
% position (old hardcoded value of .3\normalbaselineskip is far too high
% when used with very small footer fonts).
% Jan 31, 1996:
% version 1.95: call \@normalsize in the reset code if that is defined,
% otherwise \normalsize.
% this is to solve a problem with ucthesis.cls, as this doesn't
% define \@currsize. Unfortunately for latex209 calling \normalsize doesn't
% work as this is optimized to do very little, so there \@normalsize should
% be called. Hopefully this code works for all versions of LaTeX known to
% mankind.
% April 25, 1996:
% version 1.96: initialize \headwidth to a magic (negative) value to catch
% most common cases that people change it before calling \pagestyle{fancy}.
% Note it can't be initialized when reading in this file, because
% \textwidth could be changed afterwards. This is quite probable.
% We also switch to \MakeUppercase rather than \uppercase and introduce a
% \nouppercase command for use in headers. and footers.
% May 3, 1996:
% version 1.97: Two changes:
% 1. Undo the change in version 1.8 (using the pagestyle{headings} defaults
% for the chapter and section marks. The current version of amsbook and
% amsart classes don't seem to need them anymore. Moreover the standard
% latex classes don't use \markboth if twoside isn't selected, and this is
% confusing as \leftmark doesn't work as expected.
% 2. include a call to \ps@empty in ps@@fancy. This is to solve a problem
% in the amsbook and amsart classes, that make global changes to \topskip,
% which are reset in \ps@empty. Hopefully this doesn't break other things.
% May 7, 1996:
% version 1.98:
% Added % after the line  \def\nouppercase
% May 7, 1996:
% version 1.99: This is the alpha version of fancyhdr 2.0
% Introduced the new commands \fancyhead, \fancyfoot, and \fancyhf.
% Changed \headrulewidth, \footrulewidth, \footruleskip to
% macros rather than length parameters, In this way they can be
% conditionalized and they don't consume length registers. There is no need
% to have them as length registers unless you want to do calculations with
% them, which is unlikely. Note that this may make some uses of them
% incompatible (i.e. if you have a file that uses \setlength or \xxxx=)
% May 10, 1996:
% version 1.99a:
% Added a few more % signs
% May 10, 1996:
% version 1.99b:
% Changed the syntax of \f@nfor to be resistent to catcode changes of :=
% Removed the [1] from the defs of \lhead etc. because the parameter is
% consumed by the \@[xy]lhead etc. macros.
% June 24, 1997:
% version 1.99c:
% corrected \nouppercase to also include the protected form of \MakeUppercase
% \global added to manipulation of \headwidth.
```

```
% \iffootnote command added.
% Some comments added about \@fancyhead and \@fancyfoot.
% Aug 24, 1998
% version 1.99d
% Changed the default \ps@empty to \ps@@empty in order to allow
% \fancypagestyle{empty} redefinition.
% Oct 11, 2000
% version 2.0
% Added LPPL license clause.
%
% A check for \headheight is added. An errormessage is given (once) if the
% header is too large. Empty headers don't generate the error even if
% \headheight is very small or even 0pt.
% Warning added for the use of 'E' option when twoside option is not used.
% In this case the 'E' fields will never be used.
%
% Mar 10, 2002
% version 2.1beta
% New command: \fancyhfoffset[place]{length}
% defines offsets to be applied to the header/footer to let it stick into
% the margins (if length > 0).
% place is like in fancyhead, except that only E,O,L,R can be used.
% This replaces the old calculation based on \headwidth and the marginpar
% area.
% \headwidth will be dynamically calculated in the headers/footers when
% this is used.
%
% Mar 26, 2002
% version 2.1beta2
% \fancyhfoffset now also takes h,f as possible letters in the argument to
% allow the header and footer widths to be different.
% New commands \fancyheadoffset and \fancyfootoffset added comparable to
% \fancyhead and \fancyfoot.
% Errormessages and warnings have been made more informative.
```

Appendix B

Changes implemented in fancyhdr version 4.0

Version 4 is a significant rewrite of the package

But, except for the things mentioned under the `[compatV3]` option, it should be backwards compatible with version 3.

Introduce package options: `\usepackage[`⟨`options`⟩`]{fancyhdr}`

Option `[nocheck]`. This will eliminate the check if the header/footer fits in the allocated vertical space (`\headheight` and `\footskip` resp.)

Options `[compatV3]` See below "Eliminate adjustments of `\headheight` and `\footskip`" and "Avoiding global definitions in page styling commands". I have been thinking of making the name of this option '`I-should-not-do-this-but-I-want-to-be-compatible-with-version-3`'. :)

Options `[myheadings, headings]` These redefine those page styles in terms of fancyhdr (i.e. with the rules applied). Contrary to nccfancyhdr, they will NOT become the default page style for the document. The nccfancyhdr options `[plain]` and `[empty]` have not been implemented; I think this is useless.

Eliminate adjustments of `\headheight` and `\footskip`

If the header or footer is too high (more than `\headheight` or `\footskip`, resp.), these values are no longer adjusted for the following pages. It was too confusing. There will be a warning on each page unless option [nocheck] is given. However, with the package option [compatV3] the old behaviour is kept. This should only be used as a temporary measure, not as a final solution.

The `\fancycenter` command

`\fancycenter[⟨distance⟩][⟨stretch⟩]{⟨left-mark⟩}{⟨center-mark⟩}{⟨right-mark⟩}` (taken from `nccfancyhdr`). This will fit the three parts in a box of width `\linewidth` (which will be `\headwidth` in a header). This command works like `\hbox to\linewidth{⟨left-mark⟩\hfil⟨center-mark⟩\hfil⟨right-mark⟩}` but does this more carefully trying to exactly center the central part of the text if possible. The solution for exact centering is applied if the width of ⟨`center-mark`⟩ is less than `\linewidth - 2*(⟨stretch⟩*⟨distance⟩ + max(width(⟨left-mark⟩), width(⟨right-mark⟩)))`.
Otherwise the ⟨`center-mark`⟩ will slightly migrate to a shorter item (⟨`left-mark`⟩ or ⟨`right-mark`⟩), but at least ⟨distance⟩ space between all parts of line is provided. The default values of ⟨`distance`⟩ and ⟨`stretch`⟩ are 1em and 3. If the ⟨`center-mark`⟩ is empty, then `\fancycenter` is equivalent to the following command: `\hbox to\linewidth{⟨left-mark⟩\hfil⟨right-mark⟩}`

Avoiding global definitions in page styling commands

Eliminated global definitions of headers/footers. All definitions are now local. The `\global` case was originally so that you could do definitions in a group and they would be applied globally. This was a mistake. If you make them locally they should stay local. And it caused sometimes problems with switching page styles. However, with the package option `[compatV3]` it keeps the old behaviour. This is supposed to be a temporary measure. It will disappear in the future.

`\fancypagestyle{⟨style-name⟩}[⟨base-style⟩]{⟨definitions⟩}`
The command `\fancypagestyle` gets an additional optional parameter `[⟨base-style⟩]`. Page style fancy can now also be redefined with `\fancypagestyle`.

Page style `fancydefault`

Page style `fancydefault` is page style fancy with all the defaults embedded (i.e. all the `\fancyhead`, `\fancyfoot` defs, `\headrule`, `\footrule`, `\headrulewidth`, `\footrulewidth` and the required `\chaptermark` and `\[sub]sectionmark` commands). Page style fancy does not have these embedded, it picks them up from the environment.

Parameter `\headruleskip`

This parameter changes the distance between the header text and the decorative line under it, similar to `\footruleskip`.

Commands `\fancyheadinit`, `\fancyfootinit` and `\fancyhfinit`

With `\fancyheadinit{⟨code⟩}` you can define some code that will be executed just before the construction of the header. Similarly, the code in `\fancyfootinit{⟨code⟩}` is executed in the footer. And `\fancyhfinit{⟨code⟩}` sets its code for both the header and the footer.

Changes from the nccfancyhdr comments that will not be implemented

`\headstrutheight` and `\footstrutheight`
From `nccfancyhdr`:
The distance between rules and headers/footers is controlled with the `\headstrutheight` and `\footstrutheight` commands. We insert special struts in headers and footers whose depth are calculated using the values of the mentioned commands. The defaults for both `\headstrutheight` and `\footstrutheight` are 0.3`\normalbaselineskip`. You can redefine them in just the same manner as rule width commands above. (i.e. with `\renewcommand`).
In `fancyhdr` this is implemented using `\headruleskip` and `\footruleskip`.

incorrect vertical alignment in headers leads to raising headers a bit
The problem is that the header in `fancyhdr` is at the same height as the traditional headers, but that includes the rule. I.e. the rule is at the same height as the text in traditional headers, which means the text is shifted up. In `nccfancyhdr` the text is aligned the same as the traditional headers, so the rule comes out lower. But that means the rule is below the area reserved for the header. In other words, `\headsep` is no longer the distance between the header and the page body, but between the header text and the page body. I don't want to change that, because it is an incompatible change.

some features introduced fancyhdr are unsafe
("a special cycle `\@forc` is introduced with the `\def` command"): no longer valid, I think. I cannot find anything wrong here. Maybe it is meant that the name `\@forc` could give a conflict, but now the name is localised.

Notes

1. `https://books.google.com/books?id=qsXcBwAAQBAJ`
2. George Grätzer, *Advances in TEX. IV. Header and footer control in LaTEX.* Notices Amer. Math. Soc. **41** (1994), 772-777, `https://server.math.umanitoba.ca/~gratzer/images/otherarticles/OA5.pdf`
3. `https://github.com/latex3/latex2e`
4. `https://github.com/latex3/latex3`
5. `https://github.com/pietvo/fancyhdr` resp. https://github.com/pietvo/multirow
6. `https://github.com/pietvo/fancyhdr-examples`
7. `https://github.com/vslavik/diff-pdf`

Pieter van Oostrum
Leidsche Rijn
Utrecht
pieter@vanoostrum.org