This Extends ExpressionEngine

Hoe een alledaags script uitgroeide tot een complexe software-industrie

Naam:	Tom van de Wetering
Studentnummer:	0447854
Begeleider:	Marianne van den Boomen
Master:	Nieuwe Media en Digitale Cultuur
Universiteit:	Universiteit Utrecht

11 september 2011

Inhoudsopgave

1.

Introductie

1.1 A Plea To EllisLab

Op 13 oktober 2010 maakte Kenny Meyers in een blogpost genaamd 'A Plea To EllisLab' een gevoel openbaar dat op dat moment leefde bij een groot aantal andere leden van de community rondom het contentmanagementsysteem (CMS) ExpressionEngine (EE). Meyers vat zijn frustratie samen in een aantal dringende verzoeken aan EllisLab: het Amerikaanse bedrijf dat opeenvolgend de softwareproducten pMachine, ExpressionEngine 1 en ExpressionEngine 2 op de markt bracht:

> "Stop licking your wounds over the EE2 release date fiasco. We get it. Nobody won. ExpressionEngine 2's release caused a lot of internal and external strife. Stop being so defensive. The Apple silence strategy works when you release high quality, excellent products that surprise everyone. You released ExpressionEngine 2 beta. Start talking." (Meyers, 2010)

De lancering van ExpressionEngine 2.0 heeft veel stof doen opwaaien onder de trouwe gebruikersschare van het CMS. De bron van het 'fiasco' ontstond in maart 2008. ExpressionEngine moest het tot op dat moment wat betreft bekendheid afleggen tegen concurrerende systemen als Wordpress, Joomla en Drupal, maar een 'sneak preview'-video van het nieuwe administratiepaneel zorgde voor aandacht van een groot aantal technologieblogs in de vorm van torenhoge verwachtingen:

> "This is no ordinary update. The new version of EE will be built on the CodeIgniter PHP development platform which will, in turn, offer endless possibilities for usability, productivity and customisation. For non-technical minded reader think of this as changing your car from a menial family hatchback to a high performance deluxe model!"[1]

EllisLab kondigde in de video aan dat het in de zomer van datzelfde jaar deze geheel vernieuwde en volgens velen veelbelovende versie van het CMS zou lanceren, maar die deadline werd niet gehaald. Keer op keer verscheen er een blogpost op het officiële ExpressionEngine-weblog waarin EllisLab zich excuseerde voor de vertraging. Het leverde cynische reacties op vanuit de community, vooral toen duidelijk werd dat de publiekelijke lancering van ExpressionEngine 2.0 op 1 december 2009 geen afgerond product maar een 'beta' betrof. Brendon Carr, sinds 2002 lid van het officiële forum, kon zijn frustratie niet langer onderdrukken: "weren't you guys 'putting the finishing touches on' ExpressionEngine 2.0 when you showed it off at SXSW 2008?".[2]

1.2 I Hear the Plea

De manier waarop er over de aankondiging en lancering van ExpressionEngine 2.0 geschreven en gediscussieerd wordt via internetforums, blogposts en nieuwsberichten

[1] Online beschikbaar via http://www.blue-dreamer.co.uk/blog/entry/expression-engine-2-sneak-preview (laatst bezocht op 10 augustus 2011)

[2] Online beschikbaar via http://expressionengine.com/forums/viewthread/140870/ (laatst bezocht op 12 augustus 2011)

biedt inzicht in een complex systeem van verwachtingen rondom de moeizame lancering van dit hedendaagse softwareproduct. Door middel van dit afstudeeronderzoek breng ik zowel de verschillende oorzaken achter deze opmerkelijk moeizame introductie, als de manier waarop de diverse actoren in woord en gebaar hebben gereageerd op (het uitblijven van) de lancering van ExpressionEngine 2.0 in kaart. Doel hiervan is inzicht te krijgen in de complexiteit van hedendaagse softwareontwikkeling en in de manieren waarop betrokkenen met deze complexiteit omgaan.

Het onderzoek is hiermee allereerst van waarde voor organisaties en individuen die te maken krijgen met, of zelfs afhankelijk zijn van, het platform ExpressionEngine. Tienduizenden webontwikkelaars zijn afhankelijk van ExpressionEngine voor de ontwikkeling van websites: een praktijk die voor hen in veel gevallen een belangrijke bron van inkomsten is. Zij maken websites voor klanten die op hun beurt afhankelijk zijn van ExpressionEngine voor het zelfstandig kunnen beheren van hun website. Het is een micro-industrie die naast deze actoren ook nog de (werknemers van) het bedrijf EllisLab zelf en de vele aanbieders van diensten en producten rondom het centrale product ExpressionEngine omvat. Al deze stakeholders zijn in meer of mindere mate gebaat bij een duurzame ontwikkeling van het product ExpressionEngine (cf. Gawer en Cusumano, 2002), en het wekt daarom geen verbazing dat alarmerende berichten over de voortgang van die ontwikkeling tot verhitte discussies leiden.

De vraag op welke manier stakeholders rondom softwaregebaseerde producten zich tot elkaar en de ontwikkeling van software verhouden, wint recent echter ook in het academische debat aan waarde. ExpressionEngine is een product waarvan de broncode door middel van ‘modules’, ‘plugins’ en ‘extensions’ uitgebreid kan worden. Deze mogelijkheid had al sinds de begindagen van het CMS tot gevolg dat een aanzienlijk deel van de ontwikkeling van het softwareproduct door externe ontwikkelaars geïnitieerd werd. De voordelen van deze vorm van gebruikersparticipatie werden de afgelopen jaren vooral in het populair-wetenschappelijke discours veelvuldig benoemd (Tapscott en Williams 2008, Howe 2008), maar wat in de ogen van mediawetenschapper Schäfer regelmatig wordt onderschat is de verhoogde complexiteit die dit model ten opzichte van traditionele productiemodellen met zich meebrengt:

> “what is often underestimated is the dynamic that the extended branches of production can develop. Software alone is already complex, but the social dynamic is of an even greater complexity. Engaging in a large community which itself is not homogeneous but diverse and consisting of a multitude of individual members not committed to a corporate policy, the company is in need of many communication platforms to facilitate debate, to communicate its own policies, and to explain its own point of view on issues such as copyright, fair use, and the collaborative and unpaid labour of its extended developers.” (Schäfer, 2008: 276)

Deze allesbehalve eenvoudig op waarde te schatten sociale dynamiek tussen softwareproducent, derde partijen en gebruikers zien we terug in de manier waarop EllisLab op het pleidooi van Kenny Meyers reageert. “A Plea To EllisLab” werd niet alleen

van commentaar voorzien door andere communityleden, maar ook door directeur Leslie Camacho in een officiële bedrijfsblogpost genaamd “I Hear the Plea”:

> “please know that (…) everyone here at EllisLab loves you guys. We really do. We come to work everyday for your benefit, because we love what we do. If your loved ones says ‘we need to talk’ then we need to talk” (Camacho, 2010a).

Het is een uitspraak die de hechte verbintenis tussen EllisLab en de ExpressionEngine-community aan de oppervlakte brengt, maar die tevens de oorsprong en specifieke structuur van deze gecompliceerde relatie in het midden laat. Met als doel inzicht te krijgen in de ontwikkeling van het netwerk van relaties rondom het softwareproduct ExpressionEngine gedurende het afgelopen decennium stel ik daarom de volgende vraag: *hoe heeft de dynamiek tussen EllisLab, het softwareproduct ExpressionEngine en verschillende derde partijen zich gedurende de afgelopen tien jaar ontwikkeld?*

De beantwoording van deze vraag start in het volgende hoofdstuk met een verkenning van het belang van deze casestudy binnen academische discussies over softwareontwikkeling en gebruikersparticipatie, waarna ik de methode zal introduceren waarmee ik ExpressionEngine en zijn sociale en fysieke extensies in kaart zal brengen.

2. Expression Engine: hybride en heterogeen

2.1 {embed = 'gebruikersparticipatie'}

Een onderzoek naar een softwareproduct dat eenvoudig uitgebreid kan worden door derden start de laatste jaren menigmaal met een analyse van hevige discussies rond het thema gebruikersparticipatie. Deze discussies hebben het (populair-)mediawetenschappelijke debat de afgelopen jaren in grote mate gedomineerd. Na enkele jaren vol utopische verhalen over de toegenomen macht van de gebruiker van digitale media en de vermoede revolutionaire kracht van het interactieve World Wide Web (meestal 'Web2.0' genoemd), lijkt dit debat een volwassen staat te bereiken. Dit is voornamelijk te danken aan het gedegen onderzoek van Schäfer (2008) en Van der Graaf (2009). Zij brachten onafhankelijk van elkaar het gecompliceerde participatiedebat in kaart, waarna ze een nieuwe, genuanceerde koers voor het debat vaststelden.

Door voort te bouwen op deze solide discoursanalyses kunnen we (een analyse van) de volgende veelvoorkomende foutieve veronderstellingen uit het vroege gebruikersparticipatiedebat vermijden:

> "a) thinking social progress is inherent to user participation
> b) assuming that participation is only explicit, community-based, and primarily intrinsically motivated
> c) neglecting the fact that participating in cultural production does not mean participating in power structures or benefiting from generated revenues
> d) neglecting how media practices in user participation are implemented into software design" (Schäfer, 2008: 76)

Schäfer gaat deze hardnekkige vooroordelen te lijf door te beargumenteren dat de vormen van gebruikersparticipatie die gepaard gaan met hedendaagse softwareproducten niet beoordeeld moeten worden als een revolutionair alternatief voor de traditionele culturele industrie, maar eerder als een extensie van die industrie. De klassieke tegenstelling producent/consument moet worden genuanceerd door middel van een perspectief dat de grenzen tussen deze tegenstelling in twijfel trekt en dat allerlei hybride verschijningsvormen van producenten, consumenten en derde partijen met elkaar verbindt:

> "participatory culture is therefore not limited to the new role of the users, but covers rather the unfolding dynamic connections between the various participants in cultural production. Consequently the romanticized narrative of consumers being promoted to producers and controlling the information age will be abandoned, because it appears rather as a great legend constituted by a popular discourse than by actual events." (Schäfer, 2008: 20)

Niet alleen kunnen we rondom software(ontwikkeling) allerlei verschillende soorten participanten met verschillende achtergronden onderscheiden, maar daarnaast is het van belang om verschillende vormen van participatie te benoemen. Schäfer onderscheidt 'implicit participation', dat verwijst naar onvermoede vormen van participatie waarbij gebruikers interacteren met ('Web 2.0'-)webapplicaties zoals Flickr

en YouTube, en 'explicit participation', waarbij hij bewuste vormen van participatiegedrag indeelt. Het bewerken van Wikipedia-lemma's en het modificeren van software kan allebei onder expliciete participatie geschaard worden, maar daaruit blijkt des te meer dat participatiepraktijken ook binnen het kader 'explicit partipation' sterk van elkaar kunnen verschillen:

> "explicit participation is heterogeneous and concerns users who range from unskilled novices to professional programmers and come from the most diverse contexts, such as paid labor, leisure, or unpaid voluntary work, and it is heterogeneous in terms of the methods used, too" (Schäfer, 2008: 85)

Het benoemen van dergelijke verschillen tussen verschijningsvormen van participatie stimuleert onderzoekers om studies te ontwerpen die niet zozeer vertrekken vanuit het begrip 'gebruikersparticipatie', maar vanuit een specifieke samenstelling van participanten en participatiepraktijken. Schäfer brengt de voordelen van deze strategie aan het voetlicht door aan elke vorm van participatie die hij onderscheidt een casus met specifieke participanten te verbinden. Dit levert een diversiteit aan verhoudingen tussen de verschillende participanten op, die Schäfer vervolgens categoriseert in de clusters 'confrontatie', 'implementatie' en 'integratie'. Met deze clusters brengt Schäfer de aanwezigheid van verscheidene verschijningsvormen van gebruikersparticipatie aan de oppervlakte. De groep 'consumenten' neemt niet simpelweg de rol van de groep 'producenten' over, maar er ontstaat keer op keer een heterogeen bestand van hybride actoren waarbij het traditionele onderscheid consument/producent moet worden ondergraven.

In tegenstelling tot Schäfer kiest Van der Graaf (2009) niet voor een categorisatie van verschillende vormen van gebruikersparticipatie, maar voor een uitgebreide casestudy van één van die verschijningsvormen: het 'firm-hosted 3d-platform' Second Life. Net als Schäfer trekt ze het vooroordeel in twijfel dat bedrijven hun traditionele machtspositie hebben moeten afstaan aan consumenten:

> "scholars have been quick to relate (…) social progress through user participation to the organisation of the media industry, where some kind of shift in the power relations between media firms and users seems to be implied rather than systematically investigated" (Van der Graaf, 2009: 52)

Een belangrijke reden die ten grondslag zou kunnen liggen aan het ophemelen van gebruikersparticipatie is het feit dat de non-profit softwareproducten Linux en Wikipedia niet alleen zijn uitgegroeid tot iconen in het populairwetenschappelijke debat, maar ook menigmaal gebruikt worden als casus in het wetenschappelijke debat. Second Life wordt niet gefaciliteerd door een non-profit organisatie, maar door een commercieel bedrijf en daarmee vult Van der Graaf een gat in het participatiedebat:

> "insufficient attention has been given to the ways users may participate on the firm-hosted platform (in contrast to not-for-profit platforms), what they may contribute, and how and with what frequency they may interact with others. On a similar note, a blind spot seems to have developed concerning the role of the firm, directing our attention

> from 'firms as producers' to 'firms as platform (or, service) providers' coinciding with a shift in legal contracts, and which, arguably, underpins the extent of user participation" (Van der Graaf, 2009: 52)

Een keuze voor de casus ExpressionEngine kan gemotiveerd worden door het pleidooi van Van der Graaf (en Schäfer) voor systematisch onderzoek naar de (machts)relaties tussen commerciële softwareproducenten en gebruikers die zich met de softwareontwikkeling van die bedrijven bemoeien. Het onderzoek laat zich daarmee eenvoudig categoriseren volgens het model van Schäfer en vergelijken met het onderzoek van Van der Graaf. ExpressionEngine is een softwareproduct dat expliciete participatie integreert, waardoor het duidelijk onderverdeeld kan worden onder de categorieën 'explicit participation' en 'integration of participation' die Schäfer benoemt. Daarnaast is ExpressionEngine net als de casus van Van der Graaf 'firm-hosted', waardoor de idee 'user participation as an extension of the cultural industry' (Schäfer) niet zozeer wordt uitgedaagd, maar als vanzelfsprekende aanname wordt vastgesteld bij de start van het onderzoek.

Dat ExpressionEngine een 'firm-hosted' product is dat expliciete gebruikersparticipatie op een zichtbare manier integreert staat in dit onderzoek dus niet ter discussie. Een kwestie die wel ter discussie staat is de vraag in hoeverre de casus ExpressionEngine zich laat vergelijken met de casus Second Life. Hoewel beide 'firm-hosted' zijn en expliciete participatie integreren, kunnen we minstens zoveel verschillen als overeenkomsten benoemen. Zo doet ExpressionEngine zich niet voor als een 3d-platform maar als een CMS. Belangrijker dan het verschil tussen de softwareproducten die in beide casussen worden betrokken is echter het verschil tussen de structuur van de casusbenaderingen. In tegenstelling tot het onderzoek van Van der Graaf brengt dit onderzoek de ontwikkeling van een softwareproduct en de ontwikkeling van daaraan verbonden participanten in kaart.[3] Dit detail maakt het onderzoek niet alleen daadwerkelijk van toegevoegde waarde en vernieuwend ten opzichte van het onderzoek van Van der Graaf, maar verandert tevens het perspectief op het belang van het indelen van participatie rond softwareproducten in categorieën. Wat dit onderzoek zal laten zien is dat ExpressionEngine ooit het daglicht zag als opensourceproduct zonder commercieel motief, waarbij de maker koos voor de implementatie van gebruikersparticipatie in plaats van integratie: een grondvest waarvan de sporen in het huidige, commerciële product nog steeds zichtbaar zijn.

Samengevat bouwt dit onderzoek voort op het fundament dat Schäfer (2008) en Van der Graaf (2009) hebben neergezet. Niet door net als hen het utopische discours te analyseren en te bekritiseren, of door de casus ExpressionEngine in te delen in de door Schäfer afgebakende participatiecategorieën, maar door te laten zien hoe dit softwareproduct door de jaren heen balanceerde tussen implementatie en integratie van participatie, tussen commercieel en non-profit, tussen 'closed source' en 'open source' en tussen vele andere categorieën en definities.

[3] Van der Graaf noemt een analyse van de ontwikkeling van Second Life als mogelijk vervolgonderzoek. Het feit dat Second Life na enkele zeer succesvolle jaren zowel uit de populariteitsmeters als het wetenschappelijke debat is verdwenen lijkt extra kracht te zetten achter dit advies.

2.2 {ExpressionEngine == "?"}

Hoewel voorgedefinieerde categorieën vaak ontoereikend zijn om gecompliceerde fenomenen te vatten, kunnen ze wel degelijk van waarde zijn wanneer we vertrouwd willen raken met een onderzoeksobject. Categorieën zijn daarmee zowel de gewenste lantaarn in de duisternis als de TomTom die zich buiten reeds gebaande paden geen raad weet en de bestuurder vraagt terug te keren naar de hoofdweg. De vraag 'wat is ExpressionEngine?' is daarom enerzijds de moeite van het stellen waard, maar tegelijkertijd zullen we rekening moeten houden met de kans dat een kort en adequaat antwoord op die vraag tot de categorie 'onmogelijkheden' zal behoren.

Al in de eerste zinnen van dit onderzoek bleek dat het lastig is om fenomenen te beschrijven zonder ze onder te verdelen door middel van categorieën. Ik deelde ExpressionEngine in bij de contentmanagementsystemen, tezamen met Drupal, Wordpress en Joomla, met als argument dat met deze systemen inderdaad de (tekstuele) content van websites beheerd kan worden. Deze omschrijving schommelt echter tussen categorisatie op basis van ontwerp (de manier waarop ExpressionEngine ontworpen is) en categorisatie op basis van praktijken (dat waarvoor ExpressionEngine gebruikt wordt). Wat gedurende dit onderzoek duidelijk wordt is dat ExpressionEngine in variërende mate wordt ontworpen als zijnde een CMS bedoeld voor het beheren van websites en tegelijkertijd in variërende mate gebruikt wordt voor het beheren van websites. Zouden we ExpressionEngine een CMS moeten noemen omdat het (op een bepaald moment) in hoge mate voor dit doel ontworpen is (ook wanneer het niet op deze manier gebruikt wordt)? Of zouden we ExpressionEngine een CMS moeten noemen omdat het (op een bepaald moment) door velen als zodoende gebruikt wordt (zonder dat het perse voor dit doel ontworpen is)? Een combinatie, waarin zowel ontwerp als praktijk in de richting van een categorisatie als CMS wijzen, lijkt de voorkeur te hebben.

Dat de combinatie van ontwerp en praktijk niet altijd als uitgangspunt wordt genomen bij het categoriseren en definiëren van fenomenen, verschijnt aan de oppervlakte wanneer we op zoek gaan naar de definitie van een andere categorie waartoe ExpressionEngine zonder twijfel behoort: software. Lev Manovich publiceerde in zijn boek *The Language of New Media* (2001) frequent geciteerde definities van digitale media en van software in het bijzonder. Volgens Manovich zijn alle digitale media uiteindelijk te reduceren tot een databasestructuur. Daarnaast stelt hij dat 'modulariteit' een van de principes is die digitale media onderscheiden van analoge media. Volgens de mediawetenschapper kunnen we de mediumspecifieke eigenschappen van software(producten) achterhalen door middel van formele analyses op het niveau van de programmeercode: een perspectief dat hij 'digital materialism' noemt.

Een formele analyse van ExpressionEngine op het niveau van de programmeercode wijst spoedig uit dat Manovich' stellingen wat betreft de structuur van digitale media ook op

dit softwareproduct van toepassing zijn. De databasestructuur blijkt op vele vlakken de kracht van het systeem. Daarnaast is de grotendeels modulaire opzet van de programmeercode van essentieel belang om het systeem in samenwerking met derde partijen te laten groeien. De vraag is echter wat deze informatie voor nieuwe kennis oplevert. Dat ExpressionEngine zowel databasevormig als modulair is was reeds bekend en met die observatie alleen onderscheidt het zich niet van Drupal, Wordpress en vele andere softwareproducten. Voor de verschillen tussen deze producten, die wel degelijk en in ruime getale aanwezig zijn, moeten we elders op zoek.

Adrian Mackenzie, die zich net als Manovich mengt in het debat rondom het onderzoeksobject software vanuit de discipline 'software studies', stelt dat een formele analyse inderdaad slechts een kant van de medaille belicht:

> "how to handle code as a material remains problematic. It can be analyzed formally (for instance, by using the principles of variability, modularity and transcoding described in Manovich 2001), but that approach may raise more problems than it solves, since it terms are themselves derived from software design. Also, any such formal analysis tends to abstract software from the practices and contexts attached to coding" (Mackenzie, 2006: 4)

Door zowel het ontwerp als praktijken te analyseren en die twee zijden met elkaar in verband te brengen krijgen we een vollediger beeld van het fenomeen ExpressionEngine.

Voordat ik het perspectief van Mackenzie op 'software theory' verder introduceer is het belangrijk om stil te staan bij het feit dat niet alleen softwareproducten, maar ook academische producten dikwijls gecategoriseerd worden. Een onderzoek dat het thema gebruikersparticipatie en de focus op slechts een softwareproduct combineert loopt het gevaar om gecategoriseerd te worden als zijnde onderdeel van het onlangs geïntroduceerde veld 'platform studies'. Initiatiefnemers Ian Bogost en Nick Montfort gebruiken de volgende definitie om het begrip platform te duiden: "if you can program it, then it's a platform. If you can't, then it's not" (2009: 4). ExpressionEngine kwalificeert zich voor deze brede definitie, net als zo ongeveer alle verschijningsvormen van software, en de wijze waarop het door de jaren heen 'programming' van derde partijen faciliteerde speelt een belangrijke rol in dit onderzoek. Bogost en Montfort pleiten voor uitgewerkte casestudies van een specifiek platform: een eis waaraan dit onderzoek ruimschoots voldoet. Diskwalificatie voor het veld 'platform studies' volgt echter wanneer Bogost en Montfort het veld verder vernauwen: "platform studies investigates the relationships between the hardware and software design of computing systems (platforms) and the creative works produced on these systems" (2009: 3). Het in verband brengen van creatieve werken met de hard- en software om die werken te produceren is een interessante focus en zou bijvoorbeeld kunnen verklaren waarom veel ExpressionEngine-eindproducten ('website-applicaties') verschillen van Wordpress-eindproducten ('weblogs'). Een antwoord op deze vraag is echter niet mijn doelstelling, wat het associëren van dit onderzoek met het veld 'platform studies' (zoals gedefinieerd door Bogost en Montfort) problematisch maakt. Het nieuwe veld is een waardevolle

toevoeging aan het mediawetenschappelijke domein en bevestigt de veranderende rol van producenten en consumenten, maar met de focus het eindproduct sluit het een vernieuwende kijk op de manier waarop platforms tot stand komen uit.

2.3 Transformerende Extensies

Softwareproducten en academische casestudies ontkomen dus niet aan associaties met gestandaardiseerde categorieën of andere fenomenen. Aan de hand van Mackenzie (2006) en Latour (1987, 1991, 1993, 1999, 2005) kunnen we dit soms ongewenste proces echter transformeren in bruikbaar onderzoeksgereedschap. Volgens Mackenzie is software per definitie een moeilijk te kennen onderzoeksobject: “compared to other logoi, software is somewhat excessive and vexed. It overflows its own context and creates new contexts” (Mackenzie, 2006: 17). Softwareproducten worden ontwikkeld en gebruikt vanuit een specifieke context, wat betekent dat er met elke nieuwe ontwikkelaar en elke nieuwe gebruiker een nieuwe context aan het softwareproduct gerelateerd wordt. Deze associaties tussen een technologisch product en een specifieke ontwikkel- of gebruikscontext lijken de aandacht af te leiden van de objectieve kenmerken van het product zelf, en zagen we aan het werk toen we suggereerden dat ExpressionEngine als CMS werd gezien, omdat het door de meerderheid als CMS gebruikt werd. De aanwezigheid van verschillende contexten gerelateerd aan ExpressionEngine en andere verschijningsvormen van software lijkt het werk van de softwareonderzoekers daarmee te compliceren, tenzij we de relatie tussen de programmeercode van het softwareproduct en specifieke ontwikkel- en gebruikscontexten als een hechte, niet te scheiden verbintenis zien.

Bruno Latour beargumenteerde in zijn essay “Technology is Society Made Durable” (1991) dat de context waarbinnen de ontwikkeling van een technologisch product zich voltrekt niet losgezien kan worden van de verschillende vormen die het product gedurende de ontwikkelingsperiode aanneemt. De context is daarmee niet langer een externe factor die slechts in de kantlijn van een analyse aan de orde komt, maar een centraal aandachtspunt waarmee we kunnen verklaren hoe een technologie zich in een bepaalde richting ontwikkelt. Inzicht in de ontwikkel- en gebruikscontexten van ExpressionEngine is daarmee niet langer een complicerende factor, maar kan verklaringen geven voor de manier waarop dit softwareproduct zich door de jaren heen ontwikkelde.

Het traceren van relaties tussen de programmeercode van ExpressionEngine en de omstandigheden waarin deze programmeercode tot stand is gekomen, kan ons antwoorden geven op de vraag waarom ExpressionEngine uit de code bestaat waaruit het bestaat. Hetzelfde geldt voor de relaties tussen de ontwikkeling van de programmeercode en de verschillende situaties waarbinnen die code door de jaren heen uitgevoerd en aangepast werd. Door zoveel mogelijk menselijke en niet-menselijke invloeden op de ontwikkeling van ExpressionEngine in kaart te brengen ontstaat in termen van Latour een ‘actor-netwerk’. Waar Latour in zijn artikel laat zien dat de grootte van de broekzakken van toeristen en hun taalvaardigheid gerelateerd kan worden aan de ontwikkeling van de vorm van een specifieke hotelkamersleutel, laat ik

zien hoe de programmeercode van ExpressionEngine op basis van tweerichtingsverkeer gerelateerd kan worden aan acties van externe ontwikkelaars en andere derde partijen.

In plaats van een macro-analyse op het niveau van contentmanagementsystemen of op het niveau van globale ontwikkelaarscommunities, is dit onderzoek eerder een analyse op micro-niveau dat het object van onderzoek slechts incidenteel met softwareproducten als Wordpress, Drupal, Linux en Android vergelijkt. Het voldoet daarmee aan een wens van Rieder en Schäfer om het begrip van het specifieke te prefereren boven een suggestief begrip van het algemene:

> "the dense entanglement between human and non-human we witness today increasingly calls for perspectives that zoom in at the micro-level and theorize not only the great developments of how "society and culture" relate to "technology", but first and foremost the increasingly hybrid everyday practices that are the content of human affairs." (Rieder en Schäfer, 2005: 2)

Het in kaart brengen van de ontwikkeling van een netwerk van actoren die een rol speelden bij de ontwikkeling van ExpressionEngine gedurende de afgelopen tien jaar, biedt uiteindelijk zowel inzicht in de ontwikkeling van dit specifieke softwareproduct van een relatief simpel programmeerscript tot een industriële ecologie, als in de ontwikkeling van de dynamiek tussen de specifieke producenten, specifieke gebruikers en specifieke derde partijen.

Het overkoepelende argument dat Latour in het artikel verdedigt is dat deze dynamiek in veel gevallen vertaald wordt in het ontwerp van het softwareproduct. Sporen van de dynamische relaties tussen EllisLab en derde partijen komen in de programmeercode terecht en zijn in het vervolg niet langer vluchtig en moeilijk grijpbaar maar duurzaam en empirisch aantoonbaar. In zijn boek *Cutting Code* (2006), dat voortbouwt op de principes van Latour's actor-netwerk-theorie, stelt Mackenzie dat programmeercode hiermee als zichtbare 'index' fungeert, door wederzijdse relaties tussen het product en andere actoren ('originators', 'recipients' en 'prototypes') aan de oppervlakte te brengen.

> "software is a neighborhood of relations whose contours trace contemporary production, communication and consumption. Code is a multivalent index of the relations running among different classes of entity: originators, prototypes and recipients. These classes might include people, situations, organizations, places, devices, habits and practices. In code and coding, relations are assembled, dismantled, bundled and dispersed within and across contexts. Such relations are inextricable from agential effects, from some asymmetry between doing and being done to. Indeed, agency is nothing without those relations" (Mackenzie, 2006: 169)

Zonder diepgaand onderzoek kunnen we relatief eenvoudig achterhalen dat ExpressionEngine zowel uit opensourcecode als private code bestaat en dat het een constante staat van verandering doormaakt. Door middel van een analyse volgens de principes van Latour wordt deze heterogeniteit en instabiliteit niet als complicerende factor maar als vertrekpunt erkend, waardoor een analyse van de specifieke samenstellingen en verklaringen voor dit heterogene ontwerp direct prioriteit heeft.

Belangrijker dan een antwoord op de vraag 'wat is ExpressionEngine?' in termen van stabiele 'black boxes' waarvan de precieze betekenis ter discussie staat, zijn dan ook omschrijvingen van de specifieke transformaties die ExpressionEngine onder invloed van specifieke actoren heeft ondergaan:

> "innovations show us that we never work in a world filled with actors to which fixed contours may be granted. It is not merely that their degree of attachment to a statement varies; their competence, and even their definition, can be transformed. These transformations undergone by actors are of crucial importance to us when we follow innovations, because they reveal that the unified actor (…) is itself an association made up of elements which can be redistributed. It is opening and closing these black boxes that, until now, have made understanding the entry points of innovations such a delicate process" (Latour, 1999: 109)

Het is belangrijk dat we tijdens het in kaart brengen van het actor-netwerk niet langer achteloos spreken in voor vanzelfsprekend aangenomen 'black boxes' zoals 'CMS', 'Open Source' of 'ExpressionEngine', omdat de specifieke definities en samenstellingen van deze begrippen per situatie waarin ze worden gebruikt kunnen verschillen: een stelling die door het aantal 'releases' van de programmeercode van ExpressionEngine wordt geïndiceerd.

2.4 Group_ID = Participerende Observator

Voordat we de keuze kunnen vastleggen voor een onderzoek vanuit de principes van de actor-network-theory, is het belangrijk om vast te stellen in hoeverre het onderzoeksobject en –doel hiervoor openstaat. Een omschrijving van de rol die verschillende menselijke en niet-menselijke actoren spelen is alleen mogelijk als die actoren zich daadwerkelijk laten omschrijven. Schäfer ondervond de tegenwerking van belangrijke actoren op het gebied van softwareontwikkeling aan den lijve:

> "methods provided by actor-network theory prove to offer a valuable insights into the actual dynamics of user activities and the crucial role of technological design. However, such a research is limited in several ways: It is not possible to identify all actors or to sufficiently follow them. One is confined to a certain extent to the 'willingness' of participants to communicate. It was almost impossible to receive statements from companies, not to mention actual interviews with corporate designers or decision makers" (Schäfer, 2008: 143)

Uit het pleidooi van Meyers blijkt dat de bereidheid van EllisLab om te communiceren te wensen overlaat, maar toch is er aan waardevolle bronnen rond ExpressionEngine allerminst gebrek. Meyers' pleidooi leverde een stortvloed aan reacties op, zowel van medewerkers van EllisLab als van andere betrokkenen. Deze uitingen zijn van essentieel belang voor het openen van black boxes rondom de ontwikkeling van ExpressionEngine en voor het kunnen relateren van uitingen of andere acties van derden aan ontwerpbeslissingen van EllisLab. Opgeteld bij de vele blogartikelen door EllisLab en gebruikers, forumdiscussies en twitterberichten blijkt ExpressionEngine daadwerkelijk een 'expression engine' wat betreft de beschikbaarheid van communicatieve uitingen rondom het product. Daarnaast wordt een analyse van de programmeercode als index

gefaciliteerd door het feit dat zowel de programmeercode van het softwareproduct als de specificaties van die programmeercode toegankelijk en leesbaar zijn. Een volledig transparant systeem blijft een illusie, maar de stelling dat het ecosysteem aan relevante bronnen waarvan EllisLab, ExpressionEngine, gebruikers en derde partijen deel uitmaken transparanter is dan de ecosystemen rondom gangbare producten, valt moeilijk te ontkennen. Ik zal concluderen dat deze open sfeer niet alleen het onderzoek bespoedigt, maar daarnaast een belangrijke actor binnen de richting van de ontwikkeling van ExpressionEngine blijkt te zijn.

Deze rijkdom aan bronnen neemt echter een luxeprobleem met zich mee. Zonder voorkennis is het ingewikkeld om individuele uitingen en andere sporen op waarde te schatten. Het officiële forum kent tienduizenden topics, de software bestaat uit tienduizenden regels code en er zijn tienduizenden pagina's blogartikelen aan ExpressionEngine gewijd. Hoe kan een onderzoeker de belangrijkste tekstuele actoren en omschrijvingen uit de berg aan informatie vissen?

Een rol als participerende observator blijkt van essentieel belang om zicht te krijgen op de specifieke dynamiek tussen EllisLab, gebruikers en derde partijen. Als parttime webontwikkelaar heb ik vanaf medio 2008 zowel de (on)mogelijkheden van (de extensies van) ExpressionEngine als de individuen die samen 'the EE community' vormen leren kennen. Wat gedurende dit onderzoek regelmatig naar voren zal komen is dat inzicht in de dynamiek tussen betrokkenen van groot belang is om het product op een intelligente en duurzame manier te gebruiken en te faciliteren voor klanten. Vanaf het moment dat ik mijn klanten heb overtuigd van de toegevoegde waarde van ExpressionEngine ten opzichte van concurrerende systemen als Wordpress en Drupal zijn zij voor minimaal enkele jaren afhankelijk van het softwareproduct. Vanaf dat moment is mijn reputatie als webontwikkelaar echter weer afhankelijk van de kracht van ExpressionEngine. Wat naar voren komt uit dit onderzoek is dat deze kracht staat en valt bij de relaties tussen producent EllisLab, webontwikkelaars zoals ik en derde partijen zoals de ontwikkelaars van extensies.

Een diepgaand inzicht in deze relaties is dus essentieel om de kracht van ExpressionEngine te kunnen verwoorden naar klanten en om de duurzaamheid van het softwaresysteem, waarvan mijn reputatie als webontwikkelaar afhankelijk is, te kunnen duiden. Het levert echter ook de benodigde voorkennis en praktijkervaring op om als onderzoeker mijn weg tussen bronnen met inheemse namen als '{exp:weblog:entries}', 'Loweblog', 'Devot-EE' en 'Solspace' te vinden. Zonder deze praktijkervaring was ExpressionEngine zelfs niet in aanmerking gekomen voor een uitverkiezing tot onderzoeksobject voor deze thesis, omdat dit middelgrote softwareproduct tot op heden ontbreekt als casus in het populaire- en wetenschappelijk discours rondom softwareontwikkeling en daarmee aan de aandacht van een typische softwarewetenschapper ontsnapt.

Dat softwareonderzoekers gebaat zijn bij praktijkervaring wordt onderkend door Matthew Fuller, die voor zijn bundel *Software Studies: a Lexicon* (2008) auteurs benaderde met zowel een achtergrond in programmeerwerk als een achtergrond in de

wetenschap. De invloed op het softwarestudiesdiscours van schrijvers met een hybride achtergrond was ook al voor 2008 aan de orde van de dag. Een softwareontwikkelaar wiens observaties van belang zijn in dit onderzoek is Eric Raymond, die zich zowel in de rol van ontwikkelaar als in de rol van opinieleider mengde in het debat rond Open Source Software. Hij beschreef twee dominante visies op softwareontwikkeling in een betoog genaamd "The Cathedral and the Bazaar" (1998), dat hij enerzijds vulde met omschrijvingen van waar hij als ontwikkelaar van een e-mailcliënt tegenaan liep en anderzijds met een visie op wat die observaties betekenen voor de toekomst van softwareontwikkeling. Deze combinatie is verfrissend in een landschap vol zeer technische informaticateksten en zeer theoretische mediawetenschappelijke lectuur: een indruk die eerder door Van den Boomen en Schäfer naar voren geschoven werd:

> "Raymond, being a hacker/programmer himself and with street credibility to boot, writes with a kind of amateur anthropological flavour, always as a participating observer" (Van den Boomen en Schäfer, 2005: 12)

Zijn visie op softwareontwikkeling is echter minstens zo waardevol als de manier waarop die visie tot stand kwam. De stijl van de 'Cathedral' staat voor software die vooraf tot in detail is uitgedacht en pas gelanceerd wordt wanneer alle bekende mankementen verholpen zijn. De stijl van de 'Bazaar' gaat uit van het principe dat het verhelpen van alle mankementen een illusie is en dat een beperkt script vol bugs na vroege lancering in samenwerking met vrijwillige participanten kan uitgroeien tot een complex maar functionerend softwaresysteem dat de concurrentie met kathedraalgebaseerde producten aankan. Wat deze populair-wetenschappelijke literatuur waardevol maakt is dat in dit geval niet het softwareproduct maar de softwareontwikkeling centraal staat. Dit mensenwerk blijkt net zo heterogeen en daarmee moeilijk te categoriseren als het product dat dit werk uiteindelijk oplevert.

Wie de ontwikkeling van ExpressionEngine van dichtbij volgt herkent zowel eigenschappen van de kathedraal als van de bazaar. Deze ontwikkeling resulteert in een al even heterogeen softwareproduct, dat van dichtbij de vorm aanneemt van allerlei verschillende producten. En deze hybride mix van mensenwerk en materiële programmeercode veroorzaakt een verscheidenheid aan kansen en uitdagingen voor de heterogene groep gebruikers en derde partijen die met zowel de code als de producenten van die code interacteren. Die alomtegenwoordige interactie, in woord en gebaar, maakt de 'EEcologie' uiteindelijk niet alleen hybride en heterogeen, maar ook complex en daarmee moeilijk te herleiden tot een kwestie van oorzaak en gevolg. Om grip te krijgen op deze complexiteit, begint deze actor-netwerk-analyse daarom bij het begin van het verhaal: de dag waarop Rick Ellis met een PHP-script begon.

3. Just another pMachine blog

3.1 Een extensie van Rick Ellis

3.1.1 Rick Ellis: een typische 'originator'?

Volgens Mackenzie verwijst 'code as index' in de richting van drie entiteiten die zich binnen software(ontwikkeling) tot elkaar verhouden. Naast de 'recipients' en 'prototypes' die ik verderop bespreek, benoemt Mackenzie het belang van de 'originators':

> "Software refers to its producers. Someone or something codes it: there is an originator. Whether the originator is a programmer, webmaster, corporation, software engineer, team, hacker or scripter, and regardless of whether the originator's existence can be forgotten, sanctified or criminalized, software originates somewhere" (2006: 14)

Mackenzie groepeert onder de originators niet alleen de uitvinders van softwareproducten, maar benoemt het potentiële belang van een ieder die regels code schrijft voor een softwareproduct. Hoewel ik die stelling niet ontkracht zal ik in dit hoofdstuk beargumenteren dat de grondleggers van een softwareproduct, ofwel degenen die de eerste regels code ontwikkelden en publiceerden, om meerdere redenen een belangrijk deel van de aandacht verdienen. In de pikorde wat betreft handelingsvermogen staan de uitvinders gedurende een lange periode bovenaan: vaak letterlijk omdat veel oprichters een rol als CEO of hoofdontwikkelaar blijven vervullen. De richting die met de eerste regels code wordt gekozen is weliswaar niet in lood gegoten, maar kunnen we wel degelijk als een erfenis zien waarmee nieuwe ontwikkelaars en werknemers op termijn moeten leren leven.

Als we afgaan op de aandacht die de uitvinders van softwareproducten krijgen in het populaire discours, dan is het argument dat ik maak allerminst een novum. De oprichters van belangrijke (software)producten groeien meer dan eens uit tot iconen van de multinationals die uit de toegevoegde waarde van deze producten kunnen ontstaan. Microsoft heeft Bill Gates en Steve Ballmer, Google Sergey Brin en Larry Page, Apple Steve Jobs en Steve Wozniack en Facebook Mark Zuckerberg. Deze individuen worden alom geroemd om de enorme rijkdom die ze met hun grotendeels softwaregebaseerde uitvindingen hebben vergaard en om de invloed die ze kunnen uitoefenen op de tientallen miljoenen actieve gebruikers van hun producten. Maar ook buiten de schijnwerpers van de pers krijgen uitvinders van waardevolle softwareproducten een iconische status toebedeeld. Drupal heeft Dries Buytaert, Wordpress Matt Mullenweg en GNU/Linux komt door de naamgeving voorlopig niet van haar associatie met Linus Torvalds af. Een vergelijking met Rick Ellis' EllisLab is snel gemaakt, maar hoe kunnen we daadwerkelijk inzicht krijgen in de rol die deze uitvinders in de loop der jaren hebben gespeeld? Ofwel: in hoeverre was Rick Ellis inwisselbaar voor iemand anders?

3.1.2 De heterogene motor van Rick Ellis

Om deze vraag te kunnen beantwoorden is het van essentieel belang om een helder beeld te krijgen van de specificiteit van de persoon in kwestie. Dit individu is echter niet de Rick Ellis van 2011, maar de Rick Ellis die in 2001 aan de ontwikkeling van pMachine, de voorloper van ExpressionEngine, begon. Deze versie van Ellis is niet alleen een jonger persoon, met minder ervaring en minder kennis, maar maakt ook deel uit van een geheel andere context. Zoals Latour (1991) opmerkt definieert deze context grotendeels de eigenschappen van 'Rick Ellis 2.001' en om die reden kunnen we de omstandigheden waarin Ellis zich gedurende de vroege softwareontwikkeling bevond niet negeren. Het zo uitgebreid mogelijk omschrijven van zoveel mogelijk eigenschappen van verschillende versies van Rick Ellis is echter geen doel op zich, maar dient als bronmateriaal dat ons helpt om deze eigenschappen te relateren aan de richting waarin het softwareproduct zich vanaf 2001 heeft ontwikkeld.

Het observeren van de 'offspring' van een typisch softwareproduct is echter in de meeste gevallen onmogelijk. Het cliché dat baanbrekende software dikwijls in garages of op zolderkamers wordt ontwikkeld is ook voor het stukje code genaamd pMachine van toepassing. De enige observatorparticipant was Ellis zelf. Het openen van het boek genaamd 'de vroege ontwikkeling van ExpressionEngine' is dan ook in handen van Ellis zelf. Hij grijpt die belangrijke taak in de vorm van enkele korte maar krachtige memoires aan:

> "I wrote pMachine, a web publishing/blogging tool, in the summer of 2001, while in Japan mixing some audio projects for Disney. At the time I was working as a recording engineer and sound designer, and in my spare time, building websites. After installing an eCommerce system written in a scripting language called Perl on one of my client's sites, I became fascinated by it. How did this text, which looked like gibberish, send computer instructions to the web server, allowing the shopping cart to run? I headed to the bookstore to find out. Later, when the trip to Japan materialized, since I was going to be there for several months, I saw it as an opportunity to work on the blogging engine I had started tinkering with in a language called PHP. When I returned home I had a working prototype, which I continued to refine until January 2002, when I released it to the public. Within a few months pMachine had been featured in several magazines, including MacWorld and MacAddict, and in a book about blogging published by McGraw/Hill. And so it was that I became a software entrepreneur." [4]

Uit het fragment blijkt dat we Ellis kunnen typeren als een originator met een achtergrond die we, voortbordurend op het vorige hoofdstuk, zowel heterogeen als 'in ontwikkeling' kunnen noemen. Ellis was zowel een professionele '(recording) engineer' en '(sound) designer' als een beginner in webontwikkeling en PHP. "Every good work of software starts by scratching a developer's personal itch" (Raymond, 1998: 1), en zo ook pMachine, dat hij ontwikkelde om aan een websitebehoefte van zangeres Nancy Sinatra te voldoen. Dit hobbyisme transformeerde echter binnen een paar maanden in de keuze voor fulltime ondernemerschap. De achtergrond van Ellis is daarmee niet te reduceren

[4] Online te vinden via http://rickellis.com/design/pmachine.html (Laatst bezocht op 9 augustus 2011)

tot een puur commerciële setting, maar ook niet tot een setting die bepaald wordt door hobbyisme. Daarnaast veranderden zijn kenmerken en de context waarvan hij deel uitmaakte in het jaar 2001 net zo snel als het softwareproduct zelf.

Van een aantal karakteristieken kunnen we de sporen meerdere keren tijdens de verdere ontwikkeling van (de dynamiek rondom) ExpressionEngine traceren:

- ➔ Ellis had zijn sporen verdiend in '(music) engineering and design'
- ➔ Ellis was een semi-professionele websiteontwikkelaar
- ➔ Ellis ontwierp pMachine vanuit een persoonlijke-/klantbehoefte
- ➔ Ellis raakte gefascineerd door de mogelijkheden van PHP
- ➔ Ellis lanceerde pMachine eerst gratis en koos na positieve feedback voor ondernemerschap

Daarmee is pMachine enerzijds tot stand gekomen als een typisch eenmansproject. Het hobbyisme dat gevoed wordt door een fascinatie voor de mogelijkheden van software zien we bij vele andere softwarelanceringen terug. Anderzijds zijn het nu juist de specifieke kenmerken van Ellis en de situatie waarin hij zich bevond die de richting van pMachine/ExpressionEngine in belangrijke mate hebben gestuurd. Ellis was geen Open Source-fanaat, zoals Linus Torvalds, Matt Mullenweg en Dries Buytaert, waardoor commercialisering van het softwareproduct niet tegengehouden werd door alomtegenwoordige ideologische invloeden. De ontwikkeling van een 'publishing system' was een 'personal itch', maar dan wel een die gemotiveerd werd uit een behoefte van een bestaande, betalende klant. Verderop in deze analyse zullen we zien dat de specifieke wensen van deze 'recipient', een zangeres met een groot aantal fans, nog steeds gerelateerd kunnen worden aan de manier waarop ExpressionEngine is ontworpen en wordt geprofileerd.

3.2 Een extensie van software(cultuur)

3.2.1 <?php class pMachine extends PHP {

Vanaf het moment dat Ellis een regel code Ellis in een tekstbestand met de extensie '.php' schreef, droeg pMachine/ExpressionEngine de erfenis van scripttaal PHP met zich mee. Het boek over PHP dat hij kocht blijkt een belangrijke actor die de totstandkoming van producten zoals ExpressionEngine faciliteert. Het is niet bekend welk specifiek boek Ellis gebruikte om de basisbeginselen van PHP onder de knie te krijgen, maar de veelgebruikte handleiding op php.net geeft een goed beeld van zowel de mogelijkheden van PHP als de wijze waarop die mogelijkheden onder de aandacht van webontwikkelaars worden gebracht.

Een specifiek lemma binnen die handleiding maakt duidelijk wat het betekent dat Ellis voor PHP als fundament van pMachine koos. pMachine kan gezien worden als een extensie van PHP en alles dat die taal met zich meebrengt. pMachine 'erft' de mogelijkheden en beperkingen van PHP op een manier die op zichzelf juist weer een 'feature' van PHP is. Dit principe wordt belichaamd door de functie 'extend', die wordt

geactiveerd middels het keyword ‘extends’. De mogelijkheden en filosofie achter deze functie worden duidelijk omschreven in de volgende handleidingpassage:

> “Often you need classes with similar variables and functions to another existing class. In fact, it is good practice to define a generic class which can be used in all your projects and adapt this class for the needs of each of your specific projects. To facilitate this, classes can be extensions of other classes. The extended or derived class has all variables and functions of the base class (this is called 'inheritance' despite the fact that nobody died) and what you add in the extended definition. It is not possible to subtract from a class, that is, to undefine any existing functions or variables. An extended class is always dependent on a single base class, that is, multiple inheritance is not supported. Classes are extended using the keyword 'extends'.”[5]

pMachine is geschreven in PHP en daarom erft het de functie ‘extend’. Het is echter de vraag of we deze functie puur technisch moeten opvatten. Naast het feit dat ‘extend’ pMachine in staat stelt om functies van PHP te erven en om zelf weer een set ‘basisfuncties’ en ‘extended classes’ te definiëren, kan de omschrijving van het idee achter deze functie de ontwikkelaar verleiden om ‘extend’ niet puur te zien als een technische mogelijkheid, maar vooral ook als ontwerpmodel. Softwareproducten erven daarmee niet alleen de technische mogelijkheden van PHP, maar in potentie ook de ontwerpfilosofie van de ‘server-side scripting language’.

Het is daarom belangrijk dat we het handelingsvermogen van de functie ‘extend’ niet afleiden uit de technische specificaties, maar uit dat wat het daadwerkelijk in gang zet. PHP en de functie ‘extend’ maken zelf ook onderdeel uit van complexe systemen waarin zowel mensen als andere stukjes technologie een rol spelen, waardoor we ‘extend’ volgens Mackenzie als instabiele black box moeten beschouwen:

> “software studies need not directly transfer terms, categories and operations from computer science. Despite its formality as rule-governed expression, code, as we will see, is an unstable volatile material. Rather than treating data structures (lists, tuples, queues, sequences, dictionaries, hashtables, etc) as the fixed categories and components found in new media objects, or treating algorithms (sorting, searching. addressing, swapping, optimizing, encoding, decoding, etc) as the abstract essence of computational processes, I use code to understand the mixture of mutability and stasis associated with software.” (Mackenzie, 2006: 6)

We moeten dus niet op zoek te gaan naar de essentie van ‘extend’, maar naar de manier waarop het zowel als fysieke mogelijkheid als in de vorm van een ontwerpfilosofie gerelateerd kan worden aan de specifieke ontwikkeling van pMachine. Ook Wordpress en Drupal gebruiken de PHP-functie ‘extend’. Toch verschilt de manier waarop pMachine/ExpressionEngine voor extensies ontworpen is van deze concurrenten. Dit laat zien dat de specificiteit zich niet in de essentie van ‘extend’ bevindt en dat we dus op zoek moeten naar een minder formalistische methode om de verschillen tussen softwareproducten te verklaren.

[5] Online te vinden via http://php.net/manual/en/keyword.extends.php (laatst bezocht op 5 augustus 2011)

Schäfer komt met een oplossing door het materiële aspect van 'extend' uit te breiden met twee procedures die in relatie staan tot deze 'affordance': 'design' en 'appropriation':

> "three procedures shape technology: Affordance, Design and Appropriation (…). This material aspect, called affordance, determines the design in the first place, before it affects the appropriation by users. Design creates its own affordances but is also subject to the affordances of the materials utilized" (Schäfer, 2006: 32)

PHP faciliteert de materiële infrastructuur voor pMachine en biedt de 'affordance' door middel van een functie als 'extend' om de code onder te verdelen in een 'core' en onderliggende 'extensies'. De manier waarop die mogelijkheid wordt vormgegeven en ingevuld is echter grotendeels in handen van de softwareontwikkelaar. Design schept zijn eigen handelingsmogelijkheden, en de specifieke manier waarop Ellis het onderscheid tussen 'core' en 'extensie' ontwierp en na verloop van tijd aanpaste blijkt dan ook minstens zo interessant als de manier waarop de makers van PHP de functie 'extend' vormgegeven hebben. Vooral als we verderop in deze thesis de derde partij erbij betrekken: de externe ontwikkelaar die dankzij de 'affordance' 'extend' en het daarop gebouwde 'design for participation' het softwareproduct op een specifieke manier kan 'approprieren', ofwel toe-eigenen.

3.2.2 Het ontstaan van een specifieke groep 'recipients'

Vanaf het moment dat Ellis zijn eerste webapplicatie publiekelijk beschikbaar stelde op zijn website, waren de ontwikkelaar zelf en zijn bestaande klantenkring niet langer de enige gebruikers van het softwareproduct. pMachine bleek al snel een behoefte te vervullen voor een grote groep websiteontwikkelaars en webmasters. Aan de memoires van Ellis kunnen we afleiden dat de vermelding in enkele (web)magazines werkte als een katalysator, met als gevolg dat Ellis toekomst zag in een rol als softwareontwikkelaar.

Het feit dat het softwareproduct vanaf de online lancering niet langer slechts een privetool was van Ellis maar praktisch gezien 'publiek bezit', transformeerde de relatie van Ellis ten opzichte van pMachine. Het feit dat duizenden individuen en organisaties met pMachine aan de haal gingen bracht met zich mee dat Ellis de ontwikkeling van pMachine niet langer zag als middel voor het kunnen bedienen van de wensen van klanten zoals Nancy Sinatra, maar als doel op zich. Toewijding aan het project veranderde stilaan van 'personal itch' tot noodzakelijke bijdrage om gebruikers tevreden te stellen. Daarmee vormden niet langer de eindgebruikers de centrale doelgroep van Ellis, maar een nieuwe groep bestaande uit handige bloggers die zonder verdere (betaalde) hulp van Ellis hun door pMachine aangedreven website konden opzetten, en tussenpersonen die dankzij pMachine hun eigen klanten van een dynamische website konden voorzien. Uit deze nieuwe verhouding en de extra tijd die het met zich meenam kwam uiteindelijk een nieuw verdienmodel en een nieuw daarmee gepaard gaand product naar voren: pMachine Pro.

Voortaan bestond 'het softwareproduct' uit het vrij beschikbare pMachine en een betaald product met extra functionaliteiten dat gepositioneerd werd als een product dat voldeed aan de wensen van webprofessionals. Het gevolg was dat er vanaf dat moment ook twee groepen gebruikers waren: pMachine-gebruikers en gebruikers van pMachine Pro. Deze nieuwe koers blijkt de kiem van de complexe relatie tussen Ellis, de onderneming EllisLab (dat toen nog simpelweg pMachine heette), de verschillende softwareversies en –producten en de verschillende groepen gebruikers.

Voortaan diende het product niet langer in de richting van 'recipient' Nancy Sinatra te wijzen, maar in de richting van deels onbekende gebruikers met een soms zeer diverse achtergrond. Mackenzie vergelijkt de relatie tussen softwareproduct en ontvanger (waaronder de digitale omgevingen waarbinnen de software wordt geïnstalleerd en uitgevoerd) met de relatie tussen een kunstwerk en kijker: "things that receive or execute code could include an environment or milieu in which code circulates and does something. Just as a painting indexes its viewers in various ways, software too points towards its recipients" (Mackenzie, 2006: 15). Ook een kunstenaar en een museum hebben een bepaald publiek voor ogen, maar het feit dat individuele 'recipients' zowel deels onbekend als onberekenbaar zijn, maakt de relatie van 'code' ten opzichte van ' recipients' uitermate veranderlijk en complex.

De splitsing van het softwareproduct leverde niet zozeer veel extra gebruikers op, maar voortaan werd de groep wel duidelijker onderverdeeld in hobbyisten en professionals. De gebruikersschare was altijd al heterogeen, maar voortaan was de diversiteit van de gebruikersschare af te leiden aan enerzijds het aantal downloads van de gratis versie, en anderzijds het aantal verkopen van de betaalde versie. Met dit onderscheid ontstond echter ook een voorzichtige breuk met een bredere websoftwarecultuur die rond 2003 aan een opmars bezig was. De 'weblogcultuur' werd gefaciliteerd door een klimaat waarin interactieve 'publishing tools' zoals pMachine vrij gebruikt konden worden. Het in 2003 opgerichte Wordpress was speciaal gemaakt voor deze bloggers, en wist mede dankzij vrije beschikbaarheid volgens opensourceprincipes miljoenen gebruikers aan zich te binden. Ellis leek zich met zijn achtergrond als professional echter te richten op een niche: de professionele blogger en webprofessional. Het was een keuze die van grote invloed is geweest op de verdere ontwikkeling van het product en bedrijf, maar die tevens niet voor zich sprak of eerdere keuzes ongedaan maakte.

Belangrijk om op te merken is dat de keuze voor betaalde Pro-versie niet noodzakelijk voort komt uit het feit dat de 'recipients' van pMachine voornamelijk op een professionele manier met webontwikkeling bezig waren. Het opensourceproduct Drupal bedient ook met name webprofessionals en Schäfer omschrijft een voorbeeld van een websoftwareproduct dat webprofessionals bedient en toch in stand gehouden wordt door een penetrante opensourceideologie:

> "In the field of web design, many developers collaborate in informal and non-monetary-based networks on a global scale to produce resources and production means that are exploited at a local level in so-called creative industries. Frameworks for building web applications such as Django written in the collaboratively developed programming

> language Python are designed under open-source principles by a community of programmers and web designers, who are actually collaborating to build the necessary tools for their daily business of programming web applications. Deeply rooted in Internet media practices, these designers are aware of the need for cooperation" (Schäfer, 2008: 80)

Er waren succesvolle voorbeelden van opensourceprojecten voorhanden en de wens van Ellis om zich te profileren als internetondernemer, aangevuld met zijn achtergrond in de muziekindustrie waar copyrightbescherming de normaalste zaak is, lijkt daarom doorslaggevend als het gaat om de vraag waarom hij niet voor enkel het doorzetten van het opensourcemodel koos. Was de bedrijfsfilosofie vanaf nu een uitgemaakte zaak voor alle participanten?

Voor wie bekend was met de commerciële achtergrond van Ellis en zijn wensen voor opstarten van onderneming lag commercialisering van pMachine in de lijn der verwachtingen. Daarnaast zal het een ieder die bekend is met de opensourcefilosofie moeten opvallen dat het transformeren van een deel van het product in een Pro-versie die niet vrij aangepast kan worden en waarvoor een licentie benodigd is, verraadt dat de makers geen volgelingen zijn van de door Raymond geprezen opensourceontwikkelaar Linus Torvalds:

> "Linus Torvalds's style of development—release early and often, delegate everything you can, be open to the point of promiscuity—came as a surprise. No quiet, reverent cathedral-building here—rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from *anyone*) out of which a coherent and stable system could seemingly emerge only by a succession of miracles." (Raymond, 1998: 1)

In tegenstelling tot Linux en andere succesvolle opensourcesoftware leek pMachine steeds meer als traditionele 'cathedral' ontwikkeld te worden: een hypothese die we achteraf kunnen bevestigen nu we weten dat Rick Ellis vanaf medio 2003 een jaar lang in alle stilte aan de ontwikkeling van ExpressionEngine 1.0 werkte. Het masterplan stond daarmee klaar: het vrij beschikbare en relatief eenvoudige 'script' pMachine zou worden vervangen door een luxe softwareproduct waarvoor professionals graag zouden willen betalen. En dat alles zou gerund worden door een gezond bedrijf met winstoogmerk.

3.2.3 De toe-eigening van pMachine

Er bestaat echter een kloof tussen masterplan en werkelijkheid, zeker als dit plan niet uitgesproken wordt. De gratis versie van pMachine paste niet geheel binnen wat Ellis met zijn producten voor ogen had, maar werd inmiddels al wel toegeëigend door gebruikers. Deze 'appropriatie' kent vele verschijningsvormen, maar software heeft enkele 'affordances' die deze toe-eigening bespoedigen:

> "users appropriate products on the fringes of the culture industry. Software is modifiable as any product, i.e. it can be changed, extended, and used in different contexts. But software can be modified, and then globally spread at very low costs. User communities meet online and engage in collectively in software development projects.

> This has an effect on all software-based products since users can suit them to their needs." (Schäfer, 2008: 18)

De vraag die vervolgens gesteld moet worden is: hoe werd pMachine anno 2003 toegeëigend door gebruikers, en welke consequenties had dat voor de koers die Ellis met zijn softwareproduct en –bedrijf wilde gaan varen?

Eerder bleek dat pMachine werd gebruikt: een simpel gegeven dat er echter wel voor verantwoordelijk was dat pMachine überhaupt kon uitgroeien van een hobbyproject tot een project van grotere omvang. Raymond omschrijft de situatie helder: "it is truly written: the best hacks start out as personal solutions to the author's everyday problems, and spread because the problem turns out to be typical for a large class of users" (1998: 10). Wat de publieke lancering met zich meebracht is dat pMachine niet alleen binnen een door Ellis gecontroleerde context werd gebruikt, maar in tal van ongecontroleerde en moeilijk voorspelbare contexten. Waar websites gebouwd met Wordpress gedurende de eerste paar jaar na de oprichting vrijwel allemaal op elkaar leken en door een getraind oog binnen een fractie van een seconde te herkennen waren als 'just another Wordpress blog', was de professionele gebruikersschare van pMachine een stuk minder tevreden met een vooropgelegd idee van hoe een website of weblog eruit moet zien.

Op het moment dat de software in een onvoorspelde context wordt gebruikt, blijkt in hoeverre de 'affordances' en het design van het product zijn ontworpen voor appropriatie. pMachine was openbaar op het niveau van de PHP-code en kon daarom aangepast worden door externe programmeurs. Het ontwerp van pMachine stond aanpassingen op het niveau van de broncode echter niet expliciet toe. Handige gebruikers konden 'hacks' gebruiken om ze vervolgens te delen met andere leden van de vroege community rondom het product. Maar deze situatie was allesbehalve ideaal en kwam voor Ellis in zekere zin onverwacht:

> "I had released my first web app, pMachine Pro, but during that first year, as people began using it and giving me feedback, I realized that pMachine had limited potential to meet the growing demands of my users, who seemed hell-bent on using it in ways I could never have imagined when I first wrote it" [6]

pMachine was niet ontworpen om aangepast te worden aan de verschillende contexten waarbinnen gebruikers het systeem wilden integreren. pMachine was misschien toch geen 'truly great tool' in termen van Raymond, want: "any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected" (1998: 4). Waar Raymond zou kiezen voor het bazaarmodel door hulp in te schakelen van vrijwillige externe ontwikkelaars om dit probleem op te lossen, koos Ellis ervoor om in alle stilte te werken aan een product dat wel voorbereid was voor een verscheidenheid aan contexten en dat derden op termijn in staat zou stellen om zelf de juiste 'fit' voor die contexten toe te voegen door middel van extensies.

[6] Online te vinden via http://rickellis.com/design/expressionengine.html (laatst bezocht op 8 augustus 2011)

“The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better” (Raymond, 1998: 4). Ellis creëerde pMachine, gebruikers brachten het terug met verbetersuggesties en Ellis keerde terug naar zijn garage om deze wensen in een gloednieuw model te verwerken: het klinkt als een sprookje in softwareland. Een situatie waarin slechts een deel van de ideeën werd verwerkt in het nieuwe product ligt echter een stuk dichter bij de waarheid dan een situatie waarin elke wens werd gehonoreerd. De roep van een deel van de gebruikers om ‘route open source’, met Linux, Django, Wordpress en Drupal als veelgenoemde voorbeelden, wordt gedurende de volledige geschiedenis van pMachine en ExpressionEngine via de officiële forums gehoord, maar Ellis koos zijn eigen route en was daarnaast uit zakelijke overwegingen meer geïnteresseerd in de wensen van bedrijven en professionals dan in de wensen van weblogamateurs.

Hoe kunnen we het handelingsvermogen van gebruikers van pMachine vervolgens omschrijven? Een idealistische suggestie van Raymond helpt ons op weg: “users are wonderful things to have, and not just because they demonstrate that you're serving a need, that you've done something right. Properly cultivated, they can become co-developers.” (Raymond, 1998: 3). De positieve feedback op pMachine was voor Ellis een teken om zijn eerste stappen in het domein van softwareontwikkeling op een professioneel niveau verder te zetten. pMachine was echter, anders dan bijvoorbeeld Wordpress, niet gebouwd voor de bijdrage van externe ontwikkelaars: niet qua ontwerp en niet qua achterliggende filosofie. Om toch om te gaan met een nieuwe situatie waarin duizenden nieuwe ‘klanten’ zich bemoeiden met de ontwikkeling van pMachine koos Ellis voor de ontwikkeling van een nieuw systeem dat enerzijds aan zijn eigen wensen voldeed en anderzijds de mogelijkheden van gebruikersparticipatie op een beheersbare manier zou tolereren. Een nieuw systeem met de nieuwe naam ExpressionEngine.

3.3 Just another blogging app?

Samengevat kunnen we de eerste jaren van de ontwikkeling van het softwareproduct dat later opgevolgd zou worden door ExpressionEngine omschrijven als een periode die op het eerste gezicht overeenkomsten vertoont met de ‘offspring’ van een softwareproduct als Wordpress, maar dat bij nadere bestudering toch op belangrijke punten afwijkt van zijn populaire neef. Wordpress is opgericht door een opensourceaanhanger wiens ‘personal itch’ het kunnen beheren van een persoonlijk hobbyistisch blog was. pMachine is ontwikkeld door een beginnende softwareprogrammeur met een achtergrond in de muziekindustrie waarin het beschermen van copyright aan de orde van de dag is. pMachine was net als Wordpress vanaf het begin gericht op de blogcultuur, maar waar Wordpress zowel de slogan ‘Just another Wordpress blog’ als de participatie van gebruikers vanaf het begin evangeliseerde, ontwikkelde pMachine zich langzaam maar zeker in een richting waarbij het credo ‘just another blogging app’ niet langer van toepassing was.

De open distributie werd deels vervangen door een licentiesysteem en ‘de blog’ was niet langer het enige gebruikersscenario. pMachine veranderde in twee jaar van experimenteel script in de richting van een professioneel, commercieel CMS, al bleek dat voorlopig meer uit de nieuwe titel ‘pMachine Pro’ dan uit de kwaliteit van de code. Kon Ellis met een nieuwe start in de vorm van ExpressionEngine de soms ongemakkelijke erfenis van pMachine(gebruikers) doen vergeten? Een voorlopig antwoord op deze vraag volgt in hoofdstuk 4.

4.

EE1.0-1.7: Extend your Universe!

4.1 De stille groei van ExpressionEngine

4.1.1 170.000 nieuwe regels code

Meer ervaring met PHP, nieuwe kennis van webontwikkeling, positieve en negatieve feedback op pMachine, voldoende programmeertijd en een toekomstvisie voor een softwareproduct en een –bedrijf: een greep uit de omstandigheden die Ellis in staat stelden om vol passie te werken aan de 100,000-regels dikke code van zijn nieuwe creatie: ExpressionEngine. De uitvinder memoreert:

> "in early 2004, after a year of intense development, I released ExpressionEngine. The app weighed in at 100,000 lines of code, which I single-handedly wrote in complete secrecy. So armed with a new reality, I built ExpressionEngine. Since its release, ExpressionEngine has become one of the most popular web publishing applications on the market, embraced particularly by web agencies, including some of the premier firms in the world." [7]

Met release bedoelt Ellis versie 1.0. Ten opzichte van pMachine Pro was dit product een stuk robuuster en getooid met enkele nieuwe specificaties die jaren later nog steeds geroemd worden, zoals de voor ExpressionEngine kenmerkende 'template engine'. Echter, "unlike the products of industry, a computer program is always tentative, never really finished or 'closed'" (Rieder en Schäfer, 2005: 1), wat duidelijk wordt als we met onze huidige ExpressionEngine-ervaring naar de functionaliteiten van ExpressionEngine 1.0 terugkijken:

> "pMachine Pro was our first stab at a publishing system. It was a neat little program, but it wasn't a strong enough horse to ride into the future with, so we spent a year quietly writing ExpressionEngine, and on a nervous morning in 2004 released it to the public. ExpressionEngine was quite lean when it first appeared; the initial public beta had 10 modules vs. 24 that our latest version has. Some of you might remember that the first version didn't even have search capability. The core application was there, but not much else; no wiki, no moblog, no APIs, no forum, no blacklist, no custom queries, no plugins, no third-party modules, and on and on." (Ellis, 2007b)

Net zoals het met een blik op de honderdduizenden applicaties van derden in Apple's AppStore moeilijk voor te stellen is dat de eerste iPhone alleen Apple-applicaties toestond, is het voor een huidige gebruiker moeilijk voor te stellen dat ExpressionEngine ooit enkel uit de code van Ellis bestond. ExpressionEngine had daarmee veel weg van een kathedraal. Een kathedraal die bij de lancering in 2004 bezoekers toeliet, maar in de jaren erna voornamelijk door de architecten zelf doorontwikkeld werd.

Ellis schakelde wel degelijk de hulp in van gebruikers om zijn product door te ontwikkelen. Niet op de opensourcewijze zoals concurrenten als Wordpress, maar door

[7] Online te vinden via http://rickellis.com/design/expressionengine.html (Laatst bezocht op 8 augustus 2011)

een van de vroege gebruikers van pMachine in dienst te nemen: Paul Burdick. ExpressionEngine 1.0 was nog niet ontworpen voor de massale participatie van externe ontwikkelaars, want de benodigde Application Programming Interfaces (API's) bestonden nog niet. Het ontwerp van EE1.0 faciliteerde dus nog geen officiële interface voor de interactie tussen de EE1.0-code en de extensies van derden. EE1.0 was echter wel beter dan pMachine ontworpen voor de participatie van een extra ontwikkelaar. Het systeem was voortaan onderverdeeld in een duidelijk afgebakende 'core' met basisfunctionaliteiten en losse modules die tot op zekere hoogte afzonderlijk uitgevoerd en ontwikkeld konden worden:

> "first, a modular system eases the task of coordination and downplays unexpected interactions. Second, modularity (particularly associated with the degree of standardization of the interface) allows firms to upgrade per module, or throughout the product life cycle. Third, modularity reduces production costs and time because, for example, different modules can be simultaneously developed" (Van der Graaf, 2009: 66)

Het gevolg van dit modulaire ontwerp was dat ExpressionEngine voortaan binnen de kathedraal een bazaar huisvestte. Met het robuuste ontwerp van de core als fundament konden Ellis en Burdick in hoog tempo aan de slag met het inrichten van themakamers in de kathedraal in de vorm van onafhankelijke modules. Hierdoor groeide ExpressionEngine 1.0 in enkele jaren niet alleen figuurlijk in de vorm van versienummers en functionaliteiten, maar ook letterlijk in de vorm van regels code:

> "Out of curiosity, Paul wrote a little shell script to count lines of code so he could compare our current 1.6 release with the first version. Turns out we've written nearly 70,000 new lines of code per year since the release of EE 1.0, and we're averaging 4 new modules a year. A glance at our Change Log will reveal a list of nearly 1000 items. Not too shabby considering that up until about a year ago it was only Paul and I doing all the coding" (Ellis, 2007b)

De tandem Ellis en Burdick was daarmee een gesmeerde machine, met als resultaat dat betalende gebruikers zich gemiddeld eens per kwartaal aan nieuwe functionaliteiten konden laven. Deze relatief snelle 'release cycle' deed het geklaag over de geslotenheid van het product grotendeels verstommen. ExpressionEngine-code was beschermd door copyright en mocht niet vrij gedistribueerd worden, maar was net als pMachine inzichtelijk zoals een opensourceproduct waardoor het nog steeds hacks toestond. Daarnaast had Ellis wel degelijk lessen van de opensourcecultuur geleerd, wat zich uitte in de efficiënte manier waarop Ellis en Burdick hun samenwerking vormgaven. Dit was van groots belang, omdat de complexiteit van softwareontwikkeling logischerwijs toeneemt naarmate er additionele ontwikkelaars bij deze toch al complexe professie betrokken worden. Om deze complexiteit beheersbaar te houden richten ontwikkelaars hun software volgens Mackenzie af op de structuur van hun ontwikkelpraktijk en andersom. De praktijk van softwareontwikkeling neemt kenmerken over van de software zelf:

> "their work process is metaphorically treated as programming, as setting up and running loops over time. Each level of nested loop takes into account different kinds of

> unpredictability, runs over different time frames and makes use of different ways of organizing communication within the developer team. A 'release' cycle will run between two and six months and involves writing stories: an 'iteration' will involve programming some stories and last one to four weeks; a 'task' will take a few days; and writing 'unit tests' and code in pairs will take a few hours" (Mackenzie, 2006: 152)

De code van ExpressionEngine was afgestemd op een toekomst waarin Ellis zijn kathedraal niet langer in zijn eentje kon ontwikkelen door middel van de distributie over onafhankelijke modules. De specifieke ontwikkelpraktijk werd vervolgens weer op deze kenmerken afgestemd: Ellis en Burdick kozen ervoor om hun werkzaamheden gedistribueerd, zonder gemeenschappelijk kantoor en met relatief weinig onderling overleg, tot een succes te brengen. Deze principes, groot geworden in de praktijk van opensourcesoftwareontwikkeling, droegen bij aan het feit dat ExpressionEngine zich wat betreft functionaliteit en kwaliteit van de code kon blijven meten met concurrenten als Wordpress en Drupal.[8]

De groei van de software tussen 2004 en 2007 was voor Ellis echter meer een noodzakelijk kwaad dan een toekomstdroom. Veel functionaliteiten werden ontwikkeld om in een behoefte van de makers te voorzien: een forum voor interactie met gebruikers en een wiki ten dienste van de gebruikershandleiding. En anders dan opensourceprediker Raymond, die de 'perpetual beta' benoemt tot heilige graal van softwareontwikkeling en innovatie, blijkt dat Ellis minder plezier beleeft aan de constante vraag om nieuwe functionaliteiten en 'bugfixes':

> "sadly, software is never finished. We software developers are destined to a life of perpetual updates. As much as I would love to proclaim ExpressionEngine finished so I can get on with other project ideas, that's simply not the reality of software. There are too many new ideas, too many emergent technologies, and too much competition, so even as the paint dries on 1.6, we're already mixing colors for our next version" (Ellis, 2007b)

4.1.2 Design for Custom Participation

Om te voldoen aan de wensen en verwachtingen van de creatieve, veeleisende gebruikers, was Ellis constant bezig met enerzijds het uitbrengen van nieuwe functionaliteiten en anderzijds het in steeds grotere mate uitbesteden van 'customization'-praktijken aan de gebruikers zelf. ExpressionEngine 1.0 was weliswaar nog niet volledig ontworpen voor de integratie van code van derden, maar introduceerde wel een functie die de roep om meer flexibiliteit inwilligde. Dit nieuwe ontwerp van de 'custom fields' was een directe reactie op feedback van de pMachine-community. De voorgedefinieerde contentvelden 'title', 'summary', 'body' en 'extended'

[8] Opensourceproducten ontwikkelen zich minder snel dan het aantal vrijwilligers dat (in potentie) bijdraagt aan de code doet vermoeden, omdat de communicatie tussen ontwikkelaars zowel tijd vergt als problemen oplevert en omdat de praktijk uitwijst dat oprichters als Matt Mullenweg (Wordpress) en Dries Buytaert (Drupal) ook in de jaren na lancering als hoofdontwikkelaars de richting van de software vrijwel zelfstandig blijven sturen. Het uitbesteden van ontwikkeltaken aan externen is een delicaat proces dat, net als in de traditionele bedrijfsvoering, veel tijd en kunde vergt (e.g. Schafer 2008 en Raymond 1999)

waren geschikt om typische weblogs mee te vullen, maar bleken een beperking voor gebruikers die het webloguniversum waren ontstegen. Ellis introduceerde daarom een 'quick fix' in pMachine Pro versie 2.0, door een drietal 'custom fields' toe te voegen aan het voorgeprogrammeerde veldenarsenaal. Met het herschreven ontwerp van ExpressionEngine 1.0 werd de 'quick fix' omgezet in een duurzame oplossing, die volgens gebruiker Brandon Kelly uitgroeide tot ExpressionEngine's meest innovatieve functie. In een kundige geschiedschrijving van de ontwikkeling van 'custom fields' omschrijft hij het vernieuwende element van ExpressionEngine tot in detail:

> "In my book, Custom Fields are ExpressionEngine's strongest feature. They're right at the core of what defines EE. (…) ExpressionEngine 1.0 was released in April 2004. With it, EllisLab built upon pMachine Pro's Custom Fields concept in two key ways:
>
> 1. Custom Fields now had a variety of new Field Types to choose from. One could be a drop-down menu, another could be a date picker, etc..
> 2. Custom Fields were no longer assigned directly to weblogs, but rather to Field Groups, which could contain *any number* of Custom Fields. Each weblog was then assigned one Field Group.
>
> These made for an extremely powerful combination, defining ExpressionEngine as a truly malleable content management system" (Kelly, 2010a)

Dankzij het nieuwe, flexibele ontwerp van 'custom fields' kon ExpressionEngine een stuk beter afgericht worden op de specifieke wensen van de gebruiker dan pMachine. Het was tevens een van de belangrijkste functies die mij in 2008 overhaalde om over te stappen van Wordpress op ExpressionEngine. Ook met Wordpress was het enigszins mogelijk om het standaardontwerp van zowel de 'front-end' (het deel dat websitebezoekers te zien krijgen) als de 'back-end' (het administratiepaneel en de achterliggende techniek en code) aan te passen zodat het product zich niet langer puur als weblog(systeem) presenteerde. Deze toe-eigening werd echter net als ten tijde van het vroege pMachine niet gefaciliteerd door de ontwerpers, wat het proces compliceerde en leidde tot onbevredigende resultaten. Van Hippel, die de relatie tussen innovatie en gebruikersparticipatie onderzocht, stelt dat deze houding van gebruikers ten opzichte van producten die niet aan specifieke wensen voldoen zich ook op metaniveau voordoet:

> "meta-analysis of market-segmentation studies suggests that users' needs for products are highly heterogeneous in many field. (…) When users' needs are heterogeneous, this strategy of "a few sizes fit all" will leave many users somewhat dissatisfied with the commercial products on offer and probably will leave some users seriously dissatisfied. (Von Hippel, 2005: 6)

De praktijk dat veel professionele webdesigners tot in detail zelf willen bepalen hoe de websites die ze met behulp van een websoftwareproduct ontwikkelen eruit komen te zien, werd het fundament van het ontwerp van ExpressionEngine 1.0. De positieve feedback die deze beslissing genereerde verleidde Ellis vervolgens om deze 'strongest

feature' te adopteren als slogan op productportal ExpressionEngine.com: "Say hello to the most flexible web publishing system you'll ever meet".

De flexibiliteit van ExpressionEngine 1.0 was dus verweven in het ontwerp van de 'custom fields', maar vormde ook de basis van enkele andere 'core features', waaronder de nieuwe 'template engine'. Met deze functie kunnen gebruikers dynamische webapplicaties ontwerpen, zonder dat daar PHP-kennis voor nodig is. De template engine respecteert de scheiding tussen content en design tot in detail, waardoor webdesigners zonder programmeerervaring toch in staat worden gesteld om complexe websites te realiseren.[9] Wat flexibel is voor de een is echter weer een beperking voor de ander, waardoor het probleem van de heterogene gebruikersgroep steeds weer aan de oppervlakte verschijnt. Ontwikkelaars met veel PHP-ervaring zullen in delicate gevallen liever direct met de onderliggende code communiceren, dan met de tussenliggende laag van de template engine. Absolute beginners zijn juist weer niet gebaat bij de flexibiliteit van de 'custom fields', omdat het ontbreken van voorgeprogrammeerde velden, ontwerpen en instellingen de leercurve steiler maakt. Het zijn in termen van Latour (1991) allemaal 'anti-programs' die het innovatieve karakter van ExpressionEngine steeds weer opnieuw frustreerden. Een deel van het probleem is inherent aan het maken van keuzes en een ander deel kan opgelost worden door middel van hacks of keuzeschermen ('anti-anti-programs'), maar een duurzaam antwoord op het probleem van de (in)flexibiliteit diende zich pas aan met de komst van 'third party extensions'.

Kelly's omschrijving van de ontwikkeling van 'custom fields' geeft inzicht in de manier waarop de behoefte aan extra flexibiliteit in eerste instantie tot uiting kwam in 'custom field hacks' en na enkele jaren officieel werd gefaciliteerd:

> "ExpressionEngine's ability to select variant Field Types was nothing short of revolutionary, but it was missing a key ingredient: a supported way for add-on developers to inject custom Field Types into the system. That's not to say people didn't find ways to do it, though. In September 2004, Arnold Jagt published a guide to hacking FCKeditor into ExpressionEngine 1.2. And in July 2005, Chris Jennings described how to hack TinyMCE into ExpressionEngine 1.3. Things got a little easier in May 2006. EllisLab released ExpressionEngine 1.4.2, which introduced 7 new extension hooks, making it possible for add-on developers to write their own 'fieldtype extensions'—extensions which added new Field Types to your arsenal, without requiring you to hack EE's system files" (Kelly, 2010a)

Het is een analyse die de verhouding van ExpressionEngine ten opzichte van het fenomeen gebruikersparticipatie expliciet maakt. Toetsen we de mate waarin pMachine/ExpressionEngine tot 2007 openstond voor participatie aan de hand van de ontwikkeling van de functie 'custom fields', dan kunnen we vier vormen van participatie door gebruikers onderscheiden:

1. Feedback op de voorgeprogrammeerde invoervelden in pMachine

[9] Zo heb ik zelf met behulp van de template engine en zonder kennis van PHP een 'custom' digitale leeromgeving kunnen ontwikkelen. Het resultaat is te vinden op mediavergelijking.nl.

2. De creatieve invulling van 'custom fields', een vorm van toe-eigening waarvoor ExpressionEngine 1.0 ontworpen was
3. De ontwikkeling en uitvoering van hacks waarmee 'custom fields' van externe ontwikkelaars ('field types') aan het systeem toegevoegd konden worden
4. De ontwikkeling van de officiële 'field types' die ExpressionEngine vanaf versie 1.4.2 toestond

Elk van deze vormen kunnen we in termen van Schäfer beschouwen als een vorm van expliciete participatie, tenzij Ellis ook kon achterhalen hoe pMachine gebruikt werd zonder dat gebruikers daar expliciet feedback over gaven. Er was geen gebrek aan feedback, dus waarschijnlijk was dit laatste niet nodig, waardoor het begrip 'design for participation' niet op de manier zoals Schäfer dat voor ogen heeft op ExpressionEngine van toepassing lijkt. Schäfer verwijst naar centraal aangestuurde Web2.0-applicaties zoals Flickr en Facebook die, zonder dat de gebruiker zich daar bewust van is, zijn ontworpen om de interacties tussen systeem en gebruiker te kunnen exploiteren in de vorm van waardevolle feedback of andere informatie die waarde toevoegt aan het voor participatie ontworpen product. Wat we aan de hand van deze paragraaf kunnen observeren is dat een systeem als ExpressionEngine niet zozeer is ontworpen voor impliciete participatie maar voor verschillende vormen van expliciete participatie. De 'affordance' om ExpressionEngine aan te passen bestond vanwege de open broncode altijd al, maar door middel van ontwerpbeslissingen ontwikkelde ExpressionEngine zich door de jaren heen in een product dat steeds meer op maat gemaakte vormen van expliciete participatie ging faciliteren.

4.2 De opkomst van de EEndustrie

4.2.1 Professionele herinrichting: van pMachine naar EllisLab

Langzaam maar zeker ontwikkelde het experimentele PHP-script van Rick Ellis zich in een professioneel, 'enterprise' softwareproduct. Een richting die werd ingeslagen met een professioneel businessmodel (ingeluid door de introductie van pMachine Pro) en een professioneel softwareproduct (ingeluid door de introductie van ExpressionEngine 1.0) werd gecontinueerd door middel van de naamsverandering van pMachine in EllisLab in 2007:

> "Six years ago we released our first product, a web publishing program called pMachine. At the time it seemed logical to call the company pMachine as well since our entire business was based on that product. Since then we've grown, expanded our product line, and broadened our focus. Today, EllisLab is about creating cool products, incubating bright ideas, and building relationships that help our customers succeed on the web." (Ellis, 2007c)

Ellis moest voortaan als CEO zijn aandacht verdelen over meerdere producten, waaronder het open source PHP-framework CodeIgniter en webhostingdienst EngineHosting. Daarnaast werd ExpressionEngine niet meer beheerd en ontwikkeld door slechts twee 'originators', maar door een divers team werknemers dat bestond uit

programmeurs, communitymanagers en klantenservicemedewerkers. Medewerkers werkten nog steeds vanuit huis, maar dit nam niet weg dat er salarissen betaald moesten worden en dat ExpressionEngine door (nieuwe) gebruikers niet langer werd gezien als 'dat mooie stukje software van Rick Ellis' maar als een commercieel softwareproduct dat werd gerund door een professioneel bedrijf.

Het jaar 2006 gaf Ellis het vertrouwen de stap naar verdere groei te nemen. ExpressionEngine was in een aantal jaar uitgegroeid tot en zeer volledig CMS dat steeds meer grote bedrijven en hoogaangeschreven ontwerpbureau's aan zich wist te binden. De nieuwe werknemers van het bedrijf en de gebruikers konden het product met steeds meer vertrouwen aanbevelen, wat met name zichtbaar wordt als de nieuwe 'Chief Technology Officer' Derek Jones ExpressionEngine 1.5 aankondigt:

> "There is no finger crossing at 21 pMachine Street. We of course realize that we are not infallible, as evidenced by the large list of bug fixes in this version (see the full Change Log), but 1.5 is ready. Not Google ready, not Microsoft ready, not Electronic Arts ready: it's Blizzard ready, Apple ready. Feature complete, polished, tweaked and absolutely ready for public consumption. Positioned so that the inevitable bug reports will be very limited, minor in scope, and readily squashed. The Technical Support Staff is prepared for the influx of questions about new features. Enjoy. I have; every minute of it." (Jones, 2006)

Met al deze grootspraak via website en andere media konden Ellis en zijn werknemers zich niet langer achter hobbyisme verschuilen, alleen al omdat dat grote nieuwe klanten als Apple en Sony zou afschrikken. ExpressionEngine was voortaan een professioneel product. Om de breuk van EllisLab met de amateuristische erfenis uit het pMachine-tijdperk te expliciteren werd pMachine Pro zelfs verwijderd uit de officiële 'productlijn' en vrijgegeven aan het publiek:

> "When we released ExpressionEngine in early 2004, the response far exceeded our expectations. A substantial percentage of pMachine Pro users migrated immediately to ExpressionEngine, and the number of new pM Pro users dropped drastically. This trend has continued over the past year forcing us to reevaluate whether there is a place for pMachine Pro in our product line." (Ellis, 2007c)

Betekent dit dat EllisLab voorgoed afscheid kon nemen van de heterogene cultuur van zowel professionalisme als amateurisme? Nee, want hoewel we EllisLab door de jaren heen kunnen relateren aan een toenemend aantal praktijken uit het professionele circuit, is het net zo goed mogelijk om het commerciële karakter van de organisatie en het product te nuanceren. Een voorbeeld is het feit dat EllisLab altijd geweigerd heeft om zich te laven aan het grote geld van investeringsmaatschappijen. De risico's die dit met zich mee kan brengen voor het voortbestaan van een softwareproduct waaraan vele gebruikers hun passie en leven als webontwikkelaar hadden verpand speelde een rol bij deze keuze, maar de beslissing kan net zo goed aan de persoonlijke opvatting van Ellis en Burdick gerelateerd worden. Burdick spreekt zijn afkeer uit van de miljoenen die in Web2.0-startups wordt gestoken:

> "anyone else amazed by how this area has grown in the past few years? I hesitate to use the word bubble, but it simply blows my mind the way money is being thrown around with no honest revenue coming in yet. In the back of my mind, I realize that we are perhaps not quite wild enough to enter these markets. Spending millions of dollars on new offices, snacks, meetings around the world, and courting investors is just not grounded enough for me. We like having a foundation, a sense of where we are and not simply hoping to be bought out while losing control to people who throw around money like darts hoping to hit a jackpot. Would it surprise you that none of us play the lottery either?" (Burdick, 2006)

Waarop Ellis vervolgt: "I just don't get it, which is why we've always resisted any temptation to accept funding. The road is all too littered with the carcasses of failed ventures" (Ellis, 2006a). In plaats van een keuze voor snel geld op basis van een experimenteel (idee voor een) softwareproduct, kozen de makers van ExpressionEngine al sinds het begin voor een geleidelijke ontwikkeling wat betreft software, middelen en mensen. Elke cent die uitgegeven kon worden aan nieuwe functionaliteiten moest betaald worden van eerdere inkomsten en moest het bedrijf en de gebruikers uiteindelijk ook op de langere termijn ten goede komen.

Soms zorgde het weigeren van middelen voor een groeispurt ervoor dat vernieuwingen langer op zich lieten wachten, wat hier en daar leidde tot mokkende gebruikers. Wat opvalt is dat dergelijke reacties in de forums in vrijwel alle gevallen gecounterd worden met een lofzang voor EllisLab en ExpressionEngine door loyale gebruikers. Loyaliteit aan het product en de community blijkt daarmee een actor van belang wat betreft de duurzame ontwikkeling van ExpressionEngine: een fenomeen dat volgens Schäfer niet voorbehouden is aan de casus ExpressionEngine:

> "loyalty has been described as a participatory relationship which is increasingly present in organizations with high social capital. Although it may be going too far to characterize loyalty as the driving force behind the uncoordinated but effective actions of user communities and individuals, a shared understanding of values and a common sense of defining cultural freedom formed the ideological base of these actions" (Schäfer, 2008: 172)

Uiteindelijk werden gebruikers voor wie de ontwikkeling van ExpressionEngine niet snel genoeg kon gaan steeds weer overtuigd van de door Ellis en Burdick ingegeven 'common sense' dat een duurzame, professionele ontwikkeling van ExpressionEngine uiteindelijk iedereen ten goede zou komen.

4.2.2 EllisLab: prototype voor lead users

De nieuwe naam EllisLab was vanaf 2007 niet alleen een professioneel onderkomen voor het tiental werknemers, maar ook in toenemende mate voor externe ontwikkelaars. Sinds de lancering van API's voor PHP-programmeurs in 2006 groeide het aantal extensies gestaag. Wat ik in deze paragraaf zal beargumenteren is dat de ontwikkeling van deze extensies gedurende de jaren 2006-2010 kenmerken vertoont van de ontwikkeling van ExpressionEngine van 2002-2007. Waar Ellis koos voor PHP als platform voor de lancering van een waardevol softwareproduct, kozen verscheidene

ontwikkelaars voor ExpressionEngine als platform voor de lancering van een waardevolle softwarematige extensie. Deze extensies ontwikkelden zich in veel gevallen van simpel script in een professioneel of zelfs industrieel product, waarmee de vergelijking met het centrale argument van deze thesis snel is gemaakt. Wat ik zal laten zien is dat EllisLab en ExpressionEngine tijdens de ontwikkeling van deze extensies de rol van 'prototype' aannamen.

Het 'prototype' is naast de 'originator' en 'recipient' de derde en laatste grote speler waarnaar 'code' volgens Mackenzie verwijst:

> "software sometimes resembles or represents something else, a prototype. A chess program represents, albeit not visually, a chess player. (…) Sometimes, even other software is the prototype for software (as in clones)." (Mackenzie, 2006: 16)

Voordat we ExpressionEngine kunnen vergelijken met Mackenzie's schaakspeler en voordat we extensies ontwikkeld door derde partijen kunnen vergelijken met ExpressionEngine, is het van belang om de specifieke kenmerken van de 'originators', 'recipients' en fysieke structuur van deze extensies te traceren.

ExpressionEngine was tot 2006 alleen door middel van hacks uit te breiden, waardoor de meeste 'originators' te typeren waren als gebruiker voordat ze met het ontwikkelen van extensies begonnen. Een deel van hen gebruikte hacks om het systeem naar eigen hand te zetten op punten waar het tekortschoot: bijvoorbeeld wanneer een klant om functionaliteit vroeg die ExpressionEngine zelf niet kon bieden. Gebruikers zonder PHP-achtergrond zoals ik verkopen op zo een moment 'nee' of 'bijna', door te verwijzen naar het feit dat EllisLab regelmatig versies met nieuwe features lanceert. In professionelere kringen bestaande uit ontwikkelaars met voldoende PHP-kennis is het echter vanzelfsprekend om webtools naar gelang aan te passen. Het gebruiken van dergelijke hacks was echter niet ideaal, omdat ze vaak opnieuw toegepast moeten worden bij een update van het systeem en omdat hacks door gebrek aan standaardisatie en officiële ondersteuning alleen op een omslachtige manier met minder gespecialiseerde gebruikers gedeeld konden worden. De 'recipients' van de hacks waren dan ook niet zozeer andere gebruikers van ExpressionEngine, maar de klanten van PHP-vaardige 'originators' met specifieke wensen.

Voor EllisLab was de transformatie van ExpressionEngine van een platform voor CMS-gebruikers/webdesigners in een platform voor PHP-ontwikkelaars een logische stap: gebruikers vroegen erom, concurrenten als Wordpress profiteerden van de aanwezigheid van externe plugins, en voor Ellis was het een manier om de constante vraag naar nieuwe features deels door externe ontwikkelaars te laten stillen. De introductie van API's voor extensies van externe ontwikkelaars in 2006 bracht een verandering met zich mee die we kunnen vergelijken met Apple's introductie van de App Store. Net zoals het dankzij 'jailbreaks' al mogelijk was om applicaties van derden te installeren op de eerste iPhone, was het voor 2006 mogelijk om ExpressionEngine door middel van hacks aan te passen. De 'legalisering' van extensies van derden zette echter een proces in gang dat de productie, distributie en het gebruik van deze extensies in

hoge mate bespoedigde. Externe ontwikkelaars kwamen in het bezit van behulpzame gereedschappen en handleidingen, de distributie van code van derden werd versimpeld en gestandaardiseerd, en minder ervaren gebruikers konden het systeem uitbreiden met deze code zonder dat ze zelf de 'core' van ExpressionEngine hoefden aan te passen.

Vanaf het moment dat EllisLab de eerste API's publiceerde, profileerde het ExpressionEngine steeds meer als ontwikkelplatform. Een tekst op de landingspagina voor ontwikkelaars brengt aan de oppervlakte welke voordelen EllisLab ziet in deze vorm van gebruikersparticipatie en wie daar volgens het bedrijf van profiteren. Deze wervende tekst start als volgt:

> "Are you a PHP programmer? If so, consider becoming an ExpressionEngine module developer. With its plugin, module, and extension support, ExpressionEngine is an ideal platform for PHP developers."[10]

Een analyse van de manier waarop ExpressionEngine op het niveau van de broncode was ontworpen voor deze nieuwe manier van expliciete participatie laat inderdaad zien dat ervaring met PHP benodigd is om de rol van externe ontwikkelaar te kunnen vervullen. Nadat we bij de introductie van pMachine Pro de gebruikersschare tot op zekere hoogte gesplitst zagen worden in hobbyisten en professionele gebruikers, maakte EllisLab met de introductie van het ontwikkelaarprogramma het onderscheid tussen ExpressionEngine-gebruikers en ExpressionEngine-ontwikkelaars expliciet. Het is een klein, maar belangrijk onderscheid dat zorgt voor een type spraakverwarring dat ook het gebruikersparticipatiedebat frustreerde. Gebruikers zoals ik ontwikkelen websites met behulp van ExpressionEngine, maar zijn daarmee nog geen ExpressionEngine-ontwikkelaars. Dit compliceerde met name de communicatie van EllisLab naar gebruikers, omdat het bedrijf voortaan moest specificeren welke van de twee groepen het met termen als 'the community' of 'you' precies aansprak.

Waar we eerder merkten dat een webapplicatieontwikkelaar profiteert van het platform ExpressionEngine omdat het een gebrek aan PHP-kennis kan maskeren, profiteert een PHP-moduleontwikkelaar volgens EllisLab van de 'core' die de beveiliging en andere tijdrovende programmeertaken afhandelt:

> "ExpressionEngine comes with important core resources like session and member management, security and authentication, localization, and templating, enabling you to greatly accelerate your development time by focusing only on the enhancements you need" (Ibid.)

De modulaire structuur van ExpressionEngine blijkt daarmee niet alleen van belang om extra ontwikkelaars aan het werknemersbestand toe te kunnen voegen, maar tevens om een onvermoed aantal externe PHP-ontwikkelaars te kunnen faciliteren. Zoals eerder naar voren kwam zijn de API's en andere tools speciaal ontworpen voor de groep gebruikers die ExpressionEngine voorheen door middel van hacks toe-eigende, met als

[10] Online te vinden via http://expressionengine.com/developers/ (laatst bezocht op 12 augustus 2011)

doel de nadelen van deze door gebruikers geïnitieerde vorm van appropriatie weg te nemen en te vervangen door een door EllisLab beheerd ontwerp voor participatie dat beide partijen ten goede zou moeten komen.

De laatste alinea van de wervende tekst maakt de houding van EllisLab ten opzichte van de extensieontwikkelaars expliciet:

> "since our module license is virtually unrestricted you have complete freedom to create, distribute, and even sell your own add-on modules. By becoming an ExpressionEngine Developer you'll be part of an exciting community that enjoys a large, very fast-growing user base. Your development efforts will provide ExpressionEngine users with more capability, while helping you pursue your own professional goals" (Ibid.)

Anders dan iPhone-app-ontwikkelaars hoeven ExpressionEngine-ontwikkelaars geen ontwikkelaarlicentie aan te schaffen en de inkomsten die een ontwikkelaar via ExpressionEngine als platform genereert komen niet voor 70% maar voor 100% ten goede aan de externe ontwikkelaar. ExpressionEngine waardeert de inbreng van externe ontwikkelaars daarmee op een manier die we kennen van Wordpress en Drupal, waar de aanwezigheid van extensies al eerder door gebruikers en hoofdontwikkelaars werd toegejuicht als een van de belangrijkste specificaties. Het belang van externe ontwikkelaars was duidelijk, maar wat EllisLab in de laatste zin laat doorschemeren is dat de groep ExpressionEngine-gebruikers de groep is en blijft waaraan EllisLab, liefst in samenwerking met de externe ontwikkelaars (of liever 'partners'), de hoogste prioriteit zal blijven geven.

Zoals eerder naar voren kwam zijn extensieontwikkelaars vaak ook gebruikers, waardoor ze zichzelf met waardevolle extensies van dienst kunnen zijn. Het gewenste gevolg van deze setting is dat de door EllisLab omschreven win-winsituatie ontstaat. Schäfer observeert eenzelfde setting vol gebruikers met dubbele belangen bij de expliciete participatie rondom de modificatie van de Xbox:

> "ultimately, the type of user participation presented in the case above has to be characterized as heterogeneous. Such a view refutes an image of user groups as mere hobbyists working solely in their leisure time, intrinsically motivated by their opposition to commercial production. Especially in the case of the modification of electronic consumer goods, the initial producer, hobbyists, and commercial third-party developers are closely linked, and individuals participating in this production often belong to more than just one of these groups simultaneously" (Schäfer, 2008: 177)

ExpressionEngine is voor PHP-programmeurs een tweekoppig platform. Enerzijds kunnen ze hun PHP-kennis ten ruste leggen door de voorgeprogrammeerde template engine te gebruiken en anderzijds kunnen ontwikkelaars er juist voor kiezen om hun PHP-kennis uit te buiten. De acties van deze tweekoppige gebruikers, die definitief het levenslicht zagen met de introductie van de API's, transformeerden de dynamiek tussen EllisLab, ExpressionEngine en de in twee groepen onderverdeelde gebruikers.

Een voorbeeld van zo een Januskop kan verduidelijken hoe een typische gebruiker met PHP-ervaring werd verleid tot het aannemen van de rol van extensieontwikkelaar en hoe dat proces de relaties tussen gebruiker, ExpressionEngine en EllisLab veranderde. Leidenaar Lodewijk Schutte begon als gebruiker van ExpressionEngine, maar groeide na de introductie van de API's al snel uit tot een gewaardeerde extensieontwikkelaar:

> "seven years ago, I discovered ExpressionEngine in a search for a new CMS for the company I was working for at the time. Three years later, I quit my job to become a full-time freelance developer, mainly doing EE implementations. Late 2009, I released my first commercial add-on, and today marks another milestone: the launch of my dedicated EE add-on site: gotolow.com. Turns out I enjoy working on add-ons more and more, while regular client work feels less satisfying than it used to. That's why I decided to try and put more emphasis on add-on development, which inevitably meant setting up this site. From now on, if you need an EE add-on, you'll know where to go" (Schutte, 2011)

Het ontbreken van voorzieningen voor externe ontwikkelaars dwong Schutte de eerste jaren in de rol van gebruiker. Vanaf 2006 publiceerde hij zijn eerste extensies, die voortkwamen uit de behoefte om aan de specifieke wensen van zijn klanten te kunnen voldoen. Deze kleinschalige extensies stelde hij gratis beschikbaar via zijn persoonlijke website, maar konden op positieve feedback van gebruikers rekenen. De jaren erna bleek Schutte een betrouwbare extensieontwikkelaar die feedback van zijn gebruikers verwerkte in nieuwe versies en die zo nu en dan een nieuwe extensie uitbracht. Geïnspireerd door ontwikkelaars die dezelfde route bewandelden koos Schutte in 2009 voor het uitbrengen van een betaalde extensie van hoge kwaliteit. Dit werd een succes en leidde tot de beslissing om de professionaliseringcyclus rondom extensieontwikkeling af te ronden door te kiezen voor een focus op zijn rol als externe ontwikkelaar en een nieuwe productwebsite waarop de commercialisering van zijn voormalige hobbyisme nadrukkelijk naar voren komt.

Deze commercialisering van (de extensies van) Schutte werd voorafgegaan door een vergelijkbaar initiatief van Brandon Kelly. Zijn omschrijving van de factoren die een rol speelden bij de (verdere) ontwikkeling van zijn extensie 'Playa' blijkt kenmerkend voor de ontwikkeling van vele andere extensies:

> "November, 2008. Four months since I had moved out and taken a new job. And I was under the impression that my stint with ExpressionEngine was over. I had only developed a couple websites with EE at my previous job. One of them required I build Playa, and when the site was done I released it for free as a token of thanks for all of the other free add-ons people had released. I had no intention of actively supporting or maintaining it – especially since I wasn't even using EE at the new job. But then I joined Twitter. ExpressionEngine users followed me, and I followed them back. I started seeing mentions of Playa here and there, and realized that people were actually using it. But they weren't all nice things – many were having issues. I wouldn't stand for that, so I started releasing updates" (Kelly, 2010)

Pas wanneer zijn extensie daadwerkelijk gebruikt wordt beseft Kelly dat doorontwikkeling de moeite loont. Eerst met het oog op het tevreden stellen van zijn

gebruikers, maar wanneer blijkt dat dit niet mogelijk is zonder grote tijdsinvesteringen kiest hij voor het ontwikkelen van een geheel opnieuw opgebouwde versie die voldoende waarde toevoegt om er geld voor te vragen. De positieve feedback van gebruikers op zijn producten verleidt Kelly vervolgens om te kiezen voor een toekomst als extensieontwikkelaar, wat hij expliciet maakt met een nieuwe website en een bedrijf genaamd 'Pixel and Tonic'. Zijn imago van hobbyist laat hij achter zich, mede door op zijn website de 'wij'-vorm te hanteren. De gebruikers van zijn extensies, of beter gezegd zijn 'klanten', reageren enthousiast op de ontwikkeling in reacties onder zijn verhaal, waaronder Ryan Battles: "I used to hate paying for things, using a free alternative when I could. Now that I've paid for a few of your extensions, I talk my clients into them as well because of how much easier they make common tasks. Looking forward to seeing more creations produced by Pixel and Tonic" (Kelly, 2010).

Deze twee beschrijvingen van de ontwikkeling die de individuele externe ontwikkelaars gedurende hun jaren als ExpressionEngine-gebruiker hebben doorgemaakt komen in opvallende mate overeen. Enerzijds is dit te wijten aan het feit dat het er alle schijn van heeft dat Schutte zich door Kelly heeft laten inspireren in zijn beslissingen, waardoor Kelly in de rol van prototype gerelateerd kan worden aan een groot deel van de acties van Schutte. Participanten rondom een softwareproduct beïnvloeden elkaar echter niet alleen op individueel niveau, maar beïnvloeden ook de sfeer, kenmerken en normen en waarden van de softwareomgeving als geheel, en vice versa. Dit zag Mackenzie terug bij de Unix-omgeving: "Unix was and remains not just another piece of software dating from the late 1960s; it adapts and continues a set of coding, software design and system administrative practices sometimes referred to as the Unix philosophy" (Mackenzie, 2006: 82). De ExpressionEngine-filosofie beperkt zich niet tot programmeerstandaarden, maar is met name door Ellis tot stand gebracht als een filosofie waarin de beste oplossingen ontstaan uit de wens van een klant (of een 'personal itch'), waar commercialisering wordt aangeraden en waar een professionele houding met een centrale rol voor gebruikers van het product aan de orde van de dag is. Kelly en Schutte zijn beide opgegroeid met deze principes door ExpressionEngine te gebruiken en door deel uit te maken van een omgeving waarin de opvattingen van Ellis op softwareontwikkeling worden gedeeld en waarin die opvattingen op een duurzame manier zijn verwerkt in het ontwerp van het product en het bedrijf. Uit kleine verschillen tussen de routes die Schutte en Kelly hebben afgelegd van gebruiker naar fulltime externe ontwikkelaar kunnen we opmaken dat die route per individu verschilt, maar dankzij prototypes EllisLab en ExpressionEngine komen de belangrijkste karakteristieken echter wel degelijk overeen.

4.2.3 Een ecosysteem met EllisLab als epicentrum

Met de professionaliseringslag van EllisLab en de opkomst van professionele externe ontwikkelaars begonnen de tekenen van de opkomst van een 'EEndustrie' rondom het ExpressionEngine-platform duidelijkere vormen aan te nemen. Als product maakte ExpressionEngine altijd al onderdeel uit van (miljarden)industrieën, maar langzamerhand transformeerde ExpressionEngine zelf in een industrie waar verscheidene goederen en diensten geproduceerd, verhandeld, gedistribueerd en gecombineerd werden. In de vorige twee paragrafen hebben we kunnen zien hoe deze

industrie op microniveau deels onverwacht en deels volgens een voorspelbaar patroon tot stand kwam. Vervolgens is het de vraag wat deze industrie behalve commerciële producten van EllisLab en externe ontwikkelaars nog meer met zich mee brengt en wat dat betekent voor de manier waarop de verschillende partijen zich tot elkaar verhouden.

Ellis omschrijft in 2007 de ontstane situatie met de term 'ecosysteem':

> "We find ourselves at the epicenter of an ecosystem that includes designers, developers, business people, teachers, writers, and bloggers, so for us, it's no longer only about selling software or providing hosting, it's about building an environment that fosters symbiotic relationships between ourselves, our customers, and other professionals in our community. Pragmatically it makes sense that if we can be the hub around which an amazing community grows, everyone associated with the community will benefit" (Ellis, 2007a)

Het is een omschrijving van een koerswijziging die een direct gevolg lijkt van de keuze voor het toelaten van externe ontwikkelaars. Voortaan werd de code van ExpressionEngine in toenemende uitgebreid met soms zeer innovatieve code van derden. De vragen die Ellis zichzelf vervolgens gesteld heeft zijn: in hoeverre is EllisLab verantwoordelijk voor nieuwe features en in hoeverre kan het die taak aan de uitdijende community overlaten?

Allereerst betekende de keuze voor een functie als 'hub' of 'epicentrum' dat het belangrijk was dat EllisLab die rol zo goed mogelijk kon vervullen. Het faciliteren van honderden extensies, terwijl de 'core' niet geheel afgericht is op de karakteristieken van het principe 'extensibility', was een belangrijk argument om de 'core' en daarmee de technische hubfunctie van de grond te herbouwen met uitbreiding door externe ontwikkelaars als uitgangspunt. De ontwikkeling van ExpressionEngine 2.0 werd in gang gezet, maar tegelijkertijd diende EllisLab de rol van hub ook in het tijdperk van EE1.6 naar tevredenheid te vervullen.

Het aantal en de kwaliteit van de onbetaalde en betaalde extensies nam steeds grotere vormen aan, wat gezien kan worden als een indicatie dat EllisLab de rol van hub naar wens vervulde. Een analyse van de rol die derde partijen speelden in 2008 geeft echter een andere indruk. Zij namen niet alleen een steeds groter deel van de 'featureproductie' voor hun rekening, maar hetzelfde ging gelden voor functies die juist bij EllisLab's rol van hub leken te passen. Het paradigma van een traditionele consument/producent-verhouding was omgeslagen in een sfeer waarin gebruikers ook taken gingen oppakken die weinig met het toevoegen van programmeercode te maken hadden. Niet alleen verruilden steeds meer trouwe gebruikers het forum voor de door gebruikers geïnitieerde hashtag #EECMS op Twitter, maar er ontstonden tevens tal van diensten rond het softwareproduct. Veruit de meest invloedrijke en 'hub-achtige' dienst werd 'Devot-EE.com', een zeer uitgebreide centrale marktplaats voor ExpressionEngine-extensies. Deze website doet in niets onder voor marktplaatsen zoals Apple's App Store, de Android Market en de plugin-directory van Wordpress, met als voornaamste verschil

dat dit initiatief van ExpressionEngine-gebruiker Ryan Masuga niet door de makers van het centrale softwareproduct beheerd werd.

Een situatie waarin een door een gebruiker ontwikkelde marktplaats voor uitbreidingen door de community wordt uitverkoren tot officieel distributiekanaal lijkt te zijn ontstaan in een klimaat waarin het centrale bedrijf de touwtjes van de hub niet stevig in handen heeft. De aandacht van EllisLab ging rond de ontwikkeling van een dienst als Devot-EE.com uit naar de moeizame introductie van ExpressionEngine 2.0 en interne problemen die in het volgende hoofdstuk benoemd zullen worden. Maar tegelijkertijd vaart Masuga mee op de golven van een klimaat van alomtegenwoordige gebruikersparticipatie dat EllisLab vanaf 2004 zelf in gang had gezet. Door zowel het product als het platform te ontwerpen voor participatie raakten steeds meer gebruikers overtuigd van het idee dat de sleutel tot ExpressionEngine-innovaties, in welke vorm dan ook, in handen was van de gebruikers.

4.3 Credit where credit's due?

Na enkele jaren participatie was de software-industrie rondom externe ontwikkelaars uitgegroeid tot de orde van grootte van EllisLab en ExpressionEngine zelf. Voormalige gebruikers voegen steeds meer waarde toe aan de 'core' en weten hun waren steeds beter te verkopen via door henzelf of door medegebruikers gecreëerde marktplaatsen. Tegen het jaar 2010 nemen gebruikers het heft steeds vaker in eigen hand: een tendens die zich voordoet op zowel het niveau van het CMS zelf, op het niveau van extensieontwikkeling en op het niveau van diensten die het ecosysteem gezond en draaiende moeten houden.

De vraag wie het krediet moet krijgen voor het feit dat deze situatie kon ontstaan is echter niet eenvoudig beantwoord. Ellis en zijn team zorgden met ExpressionEngine 1.0 en de latere modules die ze eraan toevoegden voor een zeer flexibel en professioneel systeem dat gebruikers expliciet uitdaagde om zelf met creatieve (web)oplossingen te komen. Daarnaast was de industriële ontwikkeling die EllisLab doormaakte een inspiratie voor vele extensieontwikkelaars die dankzij het gunstige licentiemodel voor extensies verleid werden tot de start van eigen ondernemingen. Dit deden ze in grote getale en met verve, waardoor de innovatie rondom ExpressionEngine gelijke tred bleef houden met concurrerende systemen als Drupal en Wordpress. Tot slot was de loyale groep traditionele ExpressionEngine-gebruikers onmisbaar om de duurzame voortzetting van het 'EEcosysteem' te kunnen garanderen. Zij waren bereid om de relatief kostbare licenties van enerzijds ExpressionEngine en anderzijds een groeiend aantal commerciële extensies aan te schaffen en gaven EllisLab en externe ontwikkelaars het vertrouwen en de benodigde feedback waardoor zij konden blijven innoveren.

Het flexibele ontwerp van ExpressionEngine zette de deuren voor verschillende vormen van gebruikersparticipatie gedurende de jaren steeds verder open. Hiermee vervaagden de grenzen tussen gebruiker en producent en tussen de verschillende type gebruikers. In hoeverre is Ryan Masuga, de ontwerper en beheerder van de centrale ingang naar

honderden extensies, slechts een gewone ExpressionEngine-gebruiker? En in hoeverre behoort Brandon Kelly, de man die enkele 'onmisbare' extensies publiceerde, tot de kapiteins van het ExpressionEngine-schip? Het zijn complexe vragen die niet met behulp van ontoereikende categorieën als 'gebruiker' en 'producent' beantwoord kunnen worden, maar waarop in het volgende hoofdstuk een voorlopig antwoord volgt door de verschillende acties te traceren die volgden op deze nieuwe situatie met getransformeerde participanten en veranderde belangen.

Een vraag die in 2009 ook nog niet beantwoord kon worden is welke hoofdpersonen krediet verdienen voor hun bijdragen aan het ecosysteem en wie niet of minder. Ook EllisLab, externe ontwikkelaars en gebruikers kampen met dit probleem, met als gevolg dat er rondom ExpressionEngine een sfeer ontstond waarin iedereen iedereen 'credits' geeft.

De liefdesuiting van EllisLab-president Camacho richting de community ("We love you guys, we really do") stond niet op zichzelf. Volgens Raymond is het uitdelen van bergen credits zelfs een voorwaarde om te kunnen slagen als leidinggevende in een opensourceklimaat:

> "Both the fetchmail and Linux kernel projects show that by properly rewarding the egos of many other hackers, a strong developer/coordinator can use the Internet to capture the benefits of having lots of co-developers without having a project collapse into a chaotic mess." (Raymond, 1998: 10)

Maar ook tussen ontwikkelaars onderling is het ophemelen van elkaars prestaties aan de orde van de dag. En daarnaast is er nog de zeer loyale groep gebruikers die ExpressionEngine en EllisLab vanzelfsprekend verdedigen voor klanten, maar die deze lijn doorzetten in webdiscussies over het ene versus het andere CMS.[11] Deze loftrompet wordt echter net zo hard in de omgekeerde richting geblazen, bijvoorbeeld door Brandon Kelly: "Pixel & Tonic simply wouldn't have been possible if not for your humbling loyalty and support over the last couple years. So give yourself a big pat on the back – you've earned it!" (Kelly, 2010a). En 'you' wordt, naast andere externe ontwikkelaars, ook door Schutte bedankt:

> "Credit where credit's due. First of all, I'd like to thank Erskine Design, and especially the talented Phil Swan and illusterous Greg Wood, for coming up with a fabulous design. Working with them to get the look and feel right was an absolute pleasure. Also a big thanks to Brandon Kelly for letting me use his quite superb Juniper module—it's a thing of beauty. Last but not least, a big thanks to *you*, for buying my stuff!" (Schutte, 2011)

Schutte heeft al het recht om de kopers van zijn software te bedanken. De vraag is echter of de cultuur waarin iedereen elkaar bedankt ook een vervolg krijgt wanneer een deel van de geprezen participanten de fout in gaan en daarmee het succes van anderen

[11] Ik heb mijn eigen enthousiasme voor ExpressionEngine in dit stuk zoveel mogelijk proberen te onderdrukken, maar ben mij bewust van de 'situatedness' die volgens Copier (2007) gepaard gaat met de rol participerende observator.

in gevaar brengen. Of het ‘EEcosysteem’ in tijden van een ‘relatiecrisis’ zijn duurzame en innovatieve karakter kan behouden ondervinden we aan den lijve in hoofdstuk 5.

5. Expression Engine 2.0 BETA

5.1 Het lange wachten op 2.0

5.1.1 Een sleek preview van een high performance deluxe model

Met als doel een analyse van de opkomst van ExpressionEngine 2.0 keren we terug naar 2006: het jaar dat dit versienummer voor het eerst door een medewerker van EllisLab in een blogpost genoemd werd. Het was het jaar dat ik eerder omschreef als een periode waarin EllisLab zich vol zelfvertrouwen presenteerde. De officiële blogposts van medewerker Burdick vielen op door hun open karakter en anders dan de ontwikkeling van ExpressionEngine 1.0, dat Ellis in alle stilte ontwikkelde, begon Burdick al over versie 2.0 toen de plannen zich nog in een vroeg stadium bevonden. Zijn visie op de toekomst van ExpressionEngine sluit naadloos aan op het beeld van de ontwikkeling van ExpressionEngine 1.0 tot 1.6 dat in het vorige hoofdstuk geschetst werd:

> "Everything underneath the hood of ExpressionEngine is going to be rewritten and improved. The core libraries in particular are going to be far easier for us and developers to use when writing code. While there are many bloggers still using ExpressionEngine to run their sites, developers and experienced designers are those pushing the boundaries of what ExpressionEngine can do. We are going to focus more on catering to that group and pushing what I am calling the ExpressionEngine platform. A moldable set of scripts that can run any website but also create web applications. There are examples of this already with ExpressionEngine, but it shall be easier and even more capable" (Burdick, 2006)

In 2003 werd Ellis geconfronteerd met een realiteit waarin gebruikers zelf de structuur van hun website wilden invullen, waarop hij het systeem geheel herschreef om aan deze behoefte te kunnen voldoen. In 2006 werd Ellis geconfronteerd met een realiteit waarin externe ontwikkelaars het systeem op grote schaal uitbreidden met extensies, waarop hij samen met Burdick wederom het idee opvatte om het systeem compleet te herschrijven met dit gebruikersscenario in het achterhoofd.

De visie van Ellis en Burdick op de toekomst van ExpressionEngine, die in de vorm van een geheel herschreven versie genaamd ExpressionEngine 2.0 realiteit zou moeten worden, stond als een huis. Zoals bleek uit het vorige hoofdstuk vervulden EllisLab en ExpressionEngine in een landschap met een toenemend aantal extensies en diensten van externe ontwikkelaars de rol van 'hub', waarbij innovatie eerder gericht was op het succesvol vervullen van die functie dan op het creëren van nieuwe functionaliteiten. Er leek een nieuwe rolverdeling te zijn ontstaan: externe ontwikkelaars zorgden voor innovatieve nieuwe functies, terwijl EllisLab zich op de 'core' van ExpressionEngine kon gaan richten. Deze 'core' kon alleen drastisch verbeterd worden met een complete 'rewrite', waardoor de keuze voor het compleet herschrijven van ExpressionEngine een logische vervolgstap was. Tegelijkertijd was het echter een loyaal gebaar jegens de externe ontwikkelaars, die EllisLab veel werk uit handen namen.

Het delen van deze toekomstvisie leidde echter wel tot hoge verwachtingen van gebruikers, externe ontwikkelaars en medewerkers. Voorlopig was het echter nog maar

de vraag of EllisLab aan deze verwachtingen kon voldoen, omdat het nog onduidelijk was op welke manier deze nieuwe versie tot stand zou moeten komen. Daarnaast had EllisLab zijn handen vol aan de verdere ontwikkeling van 'perpetual beta' ExpressionEngine 1.x. Zoals eerder naar voren kwam was Ellis' drijfveer het vinden van steeds weer een nieuwe creatieve uitdaging. Het was echter de vraag of het complexe 'EEcosysteem' gebaat was bij een nieuwe sprong in het diepe en daarnaast of het herschrijven van een CMS dat al 'feature-rich' was wel de creatieve kick zou opleveren waar Ellis naar op zoek was.

In 2006 zag de situatie er voor EllisLab echter nog rooskleurig uit. ExpressionEngine 1.x liep sinds de openstelling voor externe ontwikkelaars als een trein en het vooruitzicht op een nieuw systeem was ook voor medewerkers een bevestiging van het gevoel dat het softwarebedrijf afkoerste op een glorieuze toekomst. De ontwikkeling van ExpressionEngine 2.0 werd daarom ingezet, waarbij naast Ellis en Burdick ook de nieuwe medewerkers Derek Allard en Derek Jones een rol gingen spelen. Dit keer was het namelijk niet het relatief eenvoudige script pMachine dat klaarstond om herschreven te worden, maar een moloch genaamd ExpressionEngine 1.6 dat uit meer dan 170.000 regels code bestond. Ellis kon dit product niet net als bij de vorige 'rewrite' zelfstandig tot leven brengen. Ook voor vier ervaren ontwikkelaars was het echter moeilijk in te schatten hoeveel tijd het ontwikkelen van deze enorme softwarekathedraal zou gaan kosten. Het herschrijven van ExpressionEngine 1.x tot een bundel code die beter was geprepareerd voor de bijdrage van externe ontwikkelaars kostte meer tijd dan verwacht:

> "many wonder what's taking us so long with 2.0. Rewriting all of ExpressionEngine to a new architecture is a time consuming process, there's just no way around it. It's not glamorous work, and we have a very small team of developers who carry many duties in addition to programming ExpressionEngine 2.0. We knew it would be arduous when we began, but we were and remain firmly convinced that the change in architecture is the best for ExpressionEngine's future. We're setting a foundation for many many years of excellent development to come, from both first and third parties, that would have been difficult or impossible with ExpressionEngine 1.x. We readily admit, though, that we underestimated the time it would take. (Jones, 2008b)

Om ervoor te zorgen dat het proces toch bleef vorderen en beheersbaar bleef in de aanwezigheid van vier ontwikkelaars, koos EllisLab voor '*rapid prototyping*'. Dit is een methode die volgens Schäfer voor complexe, onoverzichtelijke softwareprojecten uitkomst biedt:

> "the complexity of software is increasing rapidly and that makes it always more difficult to plan a program in every detail before starting to write code. It is often impossible to foresee problems early on and plans and models have to be changed, tests have to be made, specifications have to be changed in the actual construction process. Agile methods like *extreme programming* and *rapid-prototyping* strive to make complexity more manageable and transform the top-down waterfall into a long series of iterations." (Schäfer, 2008: 6)

Ondanks de tegenvallende ontwikkelsnelheid was begin 2008 een groot deel van ExpressionEngine 2.0 af. Met behulp van een 'Sneak Preview'-video van het herontworpen beheerpaneel etaleerde EllisLab vervolgens de status van het proces, zonder daarbij precies te vermelden wat er wel en niet gereed was. Deze video gaat de geschiedenis in als de overtreffende trap van de verwachtingen die Burdick schepte met zijn blogpost in 2006. De beelden lieten een fraai vormgegeven beheersysteem zien dat de indruk wekte dat EE2.0 een grote verbetering zou zijn ten opzichte van EE1.x en dat een lancering niet meer ver weg zou zijn.

Het zicht op de qua exacte specificaties nog onbekende, maar conceptueel gezien veelbelovende mogelijkheden van deze nieuwe versie transformeerde de relatie van gebruikers en klanten van deze gebruikers ten opzichte van het product ExpressionEngine 1.x. Deze versie was niet langer het CMS met enkele onvolkomenheden waaraan gebruikers nog jaren vastzaten, maar een tijdelijk systeem dat voldeed in afwachting van EE2.0. In de zomer van 2008 liet ik mijn tot op dat moment belangrijkste klant de video van het nieuwe beheerpaneel zien, om hem vervolgens te beloven dat deze nieuwe versie het overgrote deel van zijn ergernissen wat betreft de functionaliteiten en het ontwerp van EE1.6 zou doen wegvegen. Net zoals de video mij kon overtuigen van de juistheid van mijn keuze voor ExpressionEngine, kon ik klanten overtuigen van de duurzaamheid van de systemen die ik ze had laten aanschaffen door te verwijzen naar een toekomst waarin alle problemen van het heden verleden tijd zouden zijn.[12] Het is een strategie die ook door Jay Fienberg tot uitvoering werd gebracht:

> "I know for myself, with EE 1.6, I felt like I could strongly recommend it to clients over other CMSs / engines. And, one of the selling points was the promise of EE 2—like, "you're going to be getting a platform that's viable for the next 5+ years—EE 1.6 is good now, and EE is still getting better—EE 2 is coming soon."[13]

De aanwezigheid van een duidelijk zichtbaar bewijs van de nabijheid van de introductie van versie 2.0 groeide uit tot een van de belangrijkste features van ExpressionEngine.

Een terugblik op de periode waarin ik aan de hand van de video gouden bergen beloofde, leert dat de video op meerdere niveau's een politiek geladen functie had. EllisLab gebruikte de video om de community gerust te stellen met nieuws over EE2.0, terwijl dat nieuwe product in feite nog lang niet af was. Weblogs die spraken van het in de introductie geponeerde 'high performance deluxe model' deden dit ook niet vanwege geheel legitieme overtuigingen, maar bijvoorbeeld om extra lezers te trekken. En ik liet me deels overtuigen door deze politieke trucs, maar vulde dit aan met mijn eigen politiekgemotiveerde strategie om mijn klanten tevreden te stellen. Een 'down-to-

[12] De praktijken die gepaard gingen met de nabijheid van EE2.0 vertonen overeenkomsten met de praktijken rondom mobiele communicatie die De Vries (2008) in kaart bracht. 'Nabije' toekomstdromen spelen een belangrijke rol rondom technologische innovatie, maar dergelijke onrealistische verwachtingen worden zelden volledig ingelost.

[13] Online te vinden via http://thenerdary.net/articles/entry/a_plea_to_ellislab (laatst bezocht op 17 augustus 2011)

earth' reactie op wat EllisLab in de video toonde kwam pas jaren later bovendrijven, toen gebruiker Mike Mella zijn gevoelens ten opzichte van de video deelde als reactie op Meyers' pleidooi:

> "When 2.0 was about to come out, EllisLab released a video demo of the new version. As evidence of the "level of detail" they'd put into it, Derek spent about 30 seconds talking about how the drag-handles for the fields in the Publish Form slide in and out when you click on the sidebar. I remember thinking even then: 'My clients can't get a calendar event to repeat and you're focusing on animating the drag handles?!'"[14]

De 'Sneak Preview'-video van een deel van ExpressionEngine 2.0 groeide uit tot een tastbaar teken van hoop op een softwareproduct dat de soms tegenvallende realiteit van ExpressionEngine 1.x zou kunnen doen vergeten. De enkeling die anno 2008 twijfels durfde te zetten bij de daadkracht van EllisLab, doorzag dat de preview eerder een 'sleek' karakter had dan dat het de community daadwerkelijk informeerde over de voortgang van de ontwikkeling van EE2.0. De video had ook voor andere gebruikers dan Mike Mella een teken aan de wand moeten zijn dat EllisLab haar focus verplaatste van 'feature requests' uit de community naar vernieuwingen die er vanaf een afstand aanlokkelijk uitzagen, maar waarvoor amper een concrete behoefte bestond.

5.1.2 Commercial Product + Open Source = Perfection

Een analyse van 'code' en 'recipients' kan ons helpen om daadwerkelijk inzicht te krijgen in hoe het er gedurende het jaar 2008 voorstond met de ontwikkeling van ExpressionEngine 2.0, en hoe die nieuwe versie de verhoudingen tussen EllisLab, gebruikers en externe ontwikkelaars zou gaan veranderen. Het programmeerwerk vond achter gesloten deuren plaats, maar door middel van blogposts van EllisLab-medewerkers kwam er regelmatig meer informatie naar buiten over EE2.0, het programmeerproces en het type 'recipients' die EllisLab met deze innovatie voor ogen had.

De ontwikkeling van ExpressionEngine 1.x in een platform voor enerzijds traditionele CMS-gebruikers en anderzijds externe ontwikkelaars, werd gecontinueerd met de plannen voor ExpressionEngine 2.0. ExpressionEngine transformeerde van een platform voor Januskoppige gebruikers in een systeem dat zelf de vorm van een Januskop had. Het was namelijk de bedoeling dat het door PHP-ontwikkelaars gerespecteerde PHP-framework CodeIgniter, dat Ellis enkele jaren ervoor had ontwikkeld om het proces van PHP-applicatieontwikkeling te versnellen, aan de reeds bestaande 'kop' ExpressionEngine toegevoegd zou worden. Webapplicatie ExpressionEngine moest het tot op heden stellen zonder CodeIgniter-fundering en de wens om de twee centrale maar tot op heden los van elkaar ontwikkelde EllisLab-producten onderling compatibel te maken, was een logische stap voor het bedrijf. De technologische aard van het framework verlangde echter dat ExpressionEngine op het niveau van de werking van de

[14] Online te vinden via http://thenerdary.net/articles/entry/a_plea_to_ellislab (laatst bezocht op 17 augustus 2011)

code een extensie van CodeIgniter zou worden in plaats van andersom. ExpressionEngine diende om deze technische reden compleet herschreven worden. Bleef dit uit dan zouden de twee producten van EllisLab voor altijd los van elkaar, zonder de gewenste kruisbestuiving, blijven voortbestaan, met als bijkomend gevolg dat extra regels ExpressionEngine-code nimmer in de inmiddels geliefde en tijdsbesparende CodeIgniter-stijl geschreven zouden kunnen worden. Het is de vraag of Ellis ooit aan een tweede volledige 'rewrite' van de inmiddels uit de kluiten gewassen CMS-code was overgegaan als CodeIgniter in een andere stal dan die van EllisLab stond. Maar dit neemt niet weg dat de nieuwe combinatie ook voor de tweekoppige gebruikersgroep veelbelovende vernieuwingen met zich mee bracht.

Derek Allard, van CodeIgniter-gebruiker uitgegroeid tot werknemer van EllisLab en hoofdontwikkelaar van het PHP-framework, benoemt de voordelen van de integratie van beide platforms in een officiële blogpost:

> "this is great news if you're an ExpressionEngine user, a CodeIgniter user, or both. As an ExpressionEngine developer you will have a greatly expanded community of talented developers working with you, and for you. I said during my talk, "The nerds are excited, and you should be excited that the nerds are excited". As a dyed in the wool nerd, I stand by this! And not only are the code bases merging, but I see the communities merging. CodeIgniter authors are nearly instantly qualified to write ExpressionEngine modules, extensions and plugins. The EE community will benefit from the energy and ideas of CodeIgniter-ers, and as a CodeIgniter author you've suddenly added thousands of new 'potential customers' to your list." (Allard, 2008)

Het onderscheid tussen ExpressionEngine-gebruikers en externe ontwikkelaars (in dit geval beperkt tot het deel met een CodeIgniter-achtergrond) komt wederom duidelijk naar voren. De belofte van de CodeIgniter-integratie en de daaropvolgende lancering van ExpressionEngine 2.0 blijkt daarmee een duurzame bevestiging van de communicatie tussen EllisLab en 'de community'. Ten tijde van de lancering van de API's richtte EllisLab zich nog onwennig op enerzijds CMS-gebruikers en anderzijds externe ontwikkelaars. EllisLab was al wel bedreven in het aanspreken van CodeIgniter-ontwikkelaars, dus in het vervolg konden de communicatiepraktijken richting die groep worden ingezet in de communicatie naar het ontwikkelaardeel van de ExpressionEngine-community. Dit uitte zich in officiële blogposts en forumcategorieën waarin niet langer slechts een type gebruiker werd aangesproken, maar waarin de gebruikersgroep gesplitst werd in de groep traditionele gebruikers en de groep externe (CodeIgniter-)ontwikkelaars. Het heterogene karakter van de gebruikersgroep kreeg dankzij het 'prototype' CodeIgniter-ontwikkelaar een duurzame uitwerking in de communicatiestructuren van EllisLab, maar Allard's hypothese dat er op termijn een 'merge' zou plaatsvinden tussen de CodeIgniter-community en de ExpressionEngine-community, moest op dat moment nog bewezen worden.

De multiculturele softwaresamenleving die EllisLab beoogde had niet per definitie een hoge kans van slagen, wat blijkt als we de ontstane situatie aan de hand van de vier pijlers van Mackenzie inzichtelijk proberen te maken. De programmeercode is incompatibel en moet met veel moeite herschreven worden, maar in feite geldt

hetzelfde voor de deels incompatibele 'originators', 'recipients' en 'prototypes' van CodeIgniter en ExpressionEngine 1.x. De problematiek vanuit het herschrijven van bestaande code dat uitgaat van een ander ontwikkelparadigma werd nogmaals bevestigd door Derek Jones:

> "I warned you it wasn't glamorous! In fact, some software developers declare such rewrites to be nothing short of sin. In some respects that's rather true. But the technology landscape has shifted dramatically since the design of ExpressionEngine 1.0, and to setup for great things in the future, ExpressionEngine 2.0 is the exception to the rule, deserving the rewrite and wholly new architecture. Some of the pitfalls of such an undertaking still apply though. The process is slow and tedious, code, even your own well commented code, is harder to read and decipher full intent than it is to just write new code, we cannot neglect the existing version, and our time is dictated more by what code doesn't exist yet than on what we, as developers, would enjoy working on" (Jones, 2008b)

Waar pMachine en ExpressionEngine 1.0 in relatief korte tijd ontwikkeld werden door slechts een ontwikkelaar, bleek het herschrijven van ExpressionEngine 2.0 een taak die minder creatief was en daarmee meer moeite vergde. Het betrekken van CodeIgniter-ontwikkelaar Allard bij het herschrijven van een voor hem onbekend CMS versterkte dit probleem, omdat hij tevens niet bekend was met de specifieke eisen die de andere 'originators' en de ExpressionEngine-gebruikers aan het CMS stelden. Allard werd aangenomen omdat hij bekend was met zowel CodeIgniter en CodeIgniter-ontwikkelaars, maar zoals Jones opmerkt moest bovenal de compatibiliteit met de huidige versie van ExpressionEngine 1.x en de huidige doelgroep van het CMS behouden blijven. Deze omstandigheden maakte de door Allard gewenste integratie van producten en communities voorlopig tot een illusie.

Zoals bleek uit hoofdstuk vier was het dominante prototype van ExpressionEngine in toenemende mate 'commercial', terwijl het dominante prototype van CodeIgniter 'open source' was. De belofte van integratie sprak tot de verbeelding en kreeg na verloop van tijd een duurzaam karakter in de vorm van een sprekende slogan op www.expressionengine.com:

> *"Commercial Product + Open Source = Perfection*
> *A SIMPLE MODEL WHERE EVERYBODY WINS!"*[15]

Net zoals veel andere EllisLab-slogans moet deze tekst in het licht van marketingwaarde en de door hyperbolen omgeven stijl van EllisLab worden gezien, maar de mix van commercieel product met een opensourcefundament werd ook binnen serieuzere contexten als kenmerkende feature van ExpressionEngine 2.0 benoemd. Het is echter zeer de vraag of we het debat over de juistheid van de kern van de slogan op een conceptueel niveau rond het begrip 'open source' kunnen voeren. Of een product onder de verwarrende term 'open source' valt zit hem niet alleen in de toegankelijkheid van de

[15] Online te vinden via http://www.expressionengine.com/ (laatst bezocht op 15 augustus 2011)

broncode, maar vooral in de verschillende licenties die met hedendaagse softwareproducten gepaard gaan:

> "on one level, the term "open source" refers to a certain way of handling and sharing computer software. It implies that programs are not just available in machine code, but in source code – text files written in a programming language accessible to human beings. But to qualify as open source, it is essential that the public is allowed to modify and redistribute the product." (Schäfer, 2008: 6)

De broncode van ExpressionEngine 1.x is inzichtelijk voor derden, maar verder kunnen we het product vergelijken met de manier waarop Microsoft Windows 'open' is:

> "The Windows interface technology, therefore, is 'open but not open' in that Microsoft dominated the underlying technology (the Windows source code) and drove the evolution of the interfaces, but is relatively liberal about sharing interface specifications" (Gawer en Cusumano, 2002: 13)

PHP en CodeIgniter mogen wel vrij gewijzigd en gedistribueerd worden en vallen daarmee onder 'open source', maar ExpressionEngine 2.0 zelf zou een commerciële licentie krijgen, waardoor het pakket als geheel niet langer onder 'open source' valt. Hiermee kan ExpressionEngine 2.0 letterlijk opgevat worden als een commerciële extensie van een 'open source' platform. EllisLab noemt dit de kracht van het systeem, met als argumentatie dat het profiteert van de vele vrijwillige CodeIgniter-ontwikkelaars en tegelijkertijd genoeg verdient met ExpressionEngine om 'world class support' te kunnen blijven leveren.

Het is echter maar de vraag of het nieuwe fundament veel aan de rol die opensourcesoftware al speelde rondom ExpressionEngine en EllisLab zal veranderen. Zien we ExpressionEngine niet langer als een 'black box' met een label, maar als een systeem dat bestaat uit talloze open, commerciële en halfopen elementen, dan spreekt de optelsom van ExpressionEngine en CodeIgniter ineens een stuk minder tot de verbeelding. Het complexe softwareproduct ExpressionEngine bestond namelijk altijd al uit talloze regels en ingangen voor opensourcecode: net zoals de meeste andere softwaregebaseerde systemen. Javascriptlibraries zoals jQuery bespoedigen de ontwikkeling van het interactieve beheerpaneel en door middel van extensies wordt zowel opensourcesoftware als (deels) commerciële software aan het systeem gekoppeld. Daarnaast is het weliswaar bijzonder dat CodeIgniter ook dankzij Ellis het levenslicht zag, maar dat neemt niet weg dat EllisLab (net zoals iedere CMS-ontwikkelstudio) ook voor een opensourceframework van andere makelij had kunnen kiezen.

Daarmee komt uit het deels openen van de 'black box' 'open source' naar voren dat de geschetste kruisbestuiving tussen ExpressionEngine en CodeIgniter minder realistisch is dan EllisLab ons met slogans en blogposts doet geloven. Allard's voorspelling dat CodeIgniter-ontwikkelaars massaal aan de slag zullen gaan met het ontwikkelen van extensies ten gunste van EllisLab en ExpressionEngine-gebruikers is zeer gewaagd, maar bij het uitspreken van die verwachting is het kwaad al geschied: mijn goedgelovige

klanten denken dat met EE2.0 al hun webdromen ingelost zullen worden. Ik had de term 2.0 beter nooit kunnen noemen, want vanaf dat moment ben ik het die bij uitstel of afstel van de EE2.0-revolutie op het gebrek aan ingeloste webdromen afgerekend word.

5.2 De 2.0-revolutie blijft uit

5.2.1 EECI2009: het feestje voor de storm

De revolutie van ExpressionEngine 2.0 blijft lange tijd uit:

> "all summer we've been focused on our goal of first getting the developer preview of EE 2.0 released, then the public version. The good news is that the majority of our work is done. We've had a very productive summer and accomplished a lot. Most of the major systems have been completed, and the CodeIgniter integration is functioning very well. Unfortunately, there still remain enough loose ends and details to make a summer release impossible. We're close, but as the saying goes, the devil is in the details, so we need to push the release into fall" (Ellis, 2008)

Nadat de herfstdeadline ook niet gehaald werd, volgde een lange periode waarin zowel berichten van uitstel als het ontbreken van berichten de community frustreerden. Het was een periode waarin het zicht op de kern van de vertraging werd geblokkeerd door EllisLab. De ontwikkeling vond in toenemende mate achter gesloten deuren plaats, met als doel de steeds hoger oplopende imagoschade te beperken. Dit bracht met zich mee dat de open houding van Ellis en Burdick plaatsmaakte voor blogposts die om de interne problemen binnen EllisLab heen draaiden. Wat voor ervaren softwareontwikkelaars echter allang duidelijk was, is dat EllisLab te maken had met enkele van de problemen die Mackenzie verwijzend naar een ander groot softwareproject in kaart bracht:

> "In 2001, the Deskartes Universal faced problems familiar to many software projects at the time. Frequent staffing changes, downsizing of some parts of the company, corporate mergers with other software-development companies, shifts in management direction, local aftershocks of the dotcom crash earlier in the year, and technical problems in configuring a distributed system, as well as supporting previous releases, were all affecting the project." (Mackenzie, 2006: 152)

EllisLab werkte met een klein team aan het project, waardoor het opstappen van Paul Burdick, wegens persoonlijke redenen, een grote aderlating was. Eerder bleek dat Rick Ellis minder gemotiveerd was dan tijdens de ontwikkeling van ExpressionEngine 1.0 en daarnaast had Derek Allard als CodeIgniter-ontwikkelaar te kampen met een gebrek aan 'feeling' voor het CMS ExpressionEngine. Derek Jones complementeerde het gezelschap, maar was net als andere programmeurs vooral bezig met onderhoud en de voortdurende ontwikkeling van ExpressionEngine 1.x. EllisLab communiceerde naar buiten dat het verrast werd door de omvang van het project, maar liet daarbij achterwege dat de reden dat het de problemen niet de baas kon aan andere complicerende omstandigheden lag.

De eerste Europese ExpressionEngine en CodeIgniter Conferentie (EECI2009), gehouden in Leiden, kan in meerdere opzichten gezien worden als een stilte voor de storm. Veel communityleden, waaronder ikzelf, konden de dynamiek tussen EllisLab-medewerkers, gebruikers en externe ontwikkelaars op deze conferentie voor het eerst van dichtbij ervaren. Het voelde bijzonder en vanzelfsprekend tegelijk dat de recent ontstane pikorde in woord en gebaar tot uiting kwam. EECI2009 was boven alles een viering van de ontwikkeling van ExpressionEngine 1.x tot een succesvol en geliefd platform voor extensies van externe ontwikkelaars. De kredietcultuur die ik in het vorige hoofdstuk omschreef was ook op deze conferentie alomtegenwoordig, waarbij het de medewerkers van EllisLab voorlopig vergeven werd dat de innovatie in afwachting van versie 2.0 leek te haperen. Voor velen was het een eerste fysieke ontmoeting met de 'originators' van de producten die deze mensen per saldo veel hadden opgeleverd. Speciale aandacht van zowel sprekers als toeschouwers ging echter niet uit naar EllisLab, waar Rick Ellis ontbrak, maar naar Brandon (Kelly), Leevi (Graham) en Low (Schutte): het trio externe ontwikkelaars die de afgelopen jaren met extensies (het gebrek aan) innovatie vanuit EllisLab naar de achtergrond manoeuvreerden.

Uit alles bleek dat het feestje voor de externe ontwikkelaars niet verstoord mocht worden met kritiek op EllisLab. Achteraf gezien lijkt het erop dat Veerle Pieters, een van de sprekers tijdens de tweedaagse, deze mores op een briefje heeft meegekregen. Deze getalenteerde Vlaamse webdesigner was betrokken bij de ontwikkeling van ExpressionEngine 2.0 in de rol van ontwerper van het nieuwe beheerpaneel. In haar presentatie beargumenteerde ze haar keuzes voor het specifieke ontwerp, dat anderhalf jaar eerder al in de 'sneak preview'-video te zien was. Tijdens de vragenronde en na afloop werden er grappen gemaakt over de roze kleur van een van de ontwerpversies, maar verder was er van kritiek weinig sprake. Deze kritiek, en een gedegen weerwoord, volgde pas met en na het pleidooi van Meyers, dat (hoewel pas een jaar later) als een storm op de stilte van EECI2009 volgt. Pieters pareert elke vorm van kritiek op de nieuwe layout van het EE2.0-beheerpaneel door de 'black box' van de ontwikkeling van EE2.0 te openen en voor iedereen inzichtelijk te maken. Wat blijkt is dat Pieters slechts een maand de tijd kreeg om het complexe beheerpaneel te ontwerpen. Daarnaast werd Pieters niet geconsulteerd tijdens het omzetten van ontwerp in code, met als resultaat dat de uitwerking van haar ideeën niet op haar waardering konden rekenen. Adam Khan reageert op "The Full Story of My Expression Engine 2.0 Design Involvement" en komt met de Latouriaanse gedachte dat de situatie een ander gevolg had gehad als Pieters tijdens EECI2009 haar eigen mening had gepresenteerd:

> "How different things might be today if your talk at EECI2009 had not been about what's so wonderful about EE2's new design and how we got there, but instead a comprehensive checklist of your frustrations and disappointments. Such a presentation would have been electrifying—and very possibly transformative for the product" (Pieters, 2010)

Maar zoals Brian Fidler terecht opmerkt is met Pieters' uitleg van de situatie slechts de 'black box' rondom haar rol geopend en blijft een groot deel van de ontwikkeling van ExpressionEngine nog gehuld in zwarte dozen:

> "Thanks for clarifying your role in the EE2.0 design, it's a reminder to all of us that what we see is usually only a small part of the total story"[16]

Het verhaal van de moeizame ontwikkeling van ExpressionEngine 2.0 leek er voor velen echter even niet meer toe te doen vanaf het moment dat EllisLab-manager Leslie Camacho het woord nam en dan eindelijk de definitieve 'release date' van het beloftevolle nieuwe platform aankondigde: 1 december 2009.

5.2.2 What's new?

Wat volgde was een periode van twee maanden waarin ExpressionEngine-gebruikers, en klanten van die gebruikers, eindelijk dachten te weten waar ze aan toe waren wat betreft EE2.0. Het enthousiasme en een herstellend vertrouwen in EllisLab dat daarmee gepaard ging werd echter al snel in de vorm van drie harde klappen beëindigd: ExpressionEngine 2.0 kreeg niet de status 'release candidate' maar de onbetrouwbare status 'beta':

> "The most important change is a vocab one. On 12/1 we will be releasing the ExpressionEngine 2.0 Public Beta (PB). We previously referred to it as a Release Candidate. We think calling it a Public Beta will help people better evaluate whether they should use EE 2.0 PB or EE 1.6.8 for a project." (Camacho, 2009b)

Daarnaast werd de lancering toch weer met een aantal dagen uitgesteld, maar de grootste klap moest nog komen: na meer dan twee jaar wachten was ExpressionEngine 2.0 niet het droomproduct waar de community zo lang naar had uitgekeken.

Dat het feit dat EE2.0 na drie jaar ontwikkeling nog vele oneffenheden kende was voor iedereen teleurstellend, inclusief de medewerkers van EllisLab. Volgens Raymond is dit echter kenmerkend voor dit type megaprojecten met een kathedraalstructuur:

> "In Linus's Law, I think, lies the core difference underlying the cathedral-builder and bazaar styles. In the cathedral-builder view of programming, bugs and development problems are tricky, insidious, deep phenomena. It takes months of scrutiny by a dedicated few to develop confidence that you've winkled them all out. Thus the long release intervals, and the inevitable disappointment when long-awaited releases are not perfect" (Raymond, 1999: 5)

ExpressionEngine en EllisLab hadden altijd te maken met kritiek van gebruikers over de koers en snelheid van de ontwikkeling, maar deze kritiek kwam meestal van nieuwe gebruikers en werd door loyale gebruikers gepareerd. Het forum stond kort na de

[16] Online te vinden via http://veerle.duoh.com/design/article/the_full_story_of_my_expressionengine_2_design_involvement (laatst bezocht op 17 augustus 2011)

lancering vol met negatieve recensies aan de hand van de eerste indrukken die gebruikers hadden opgedaan, maar veel opinieleiders binnen de community verbraken het patroon door deze kritiek niet te pareren, maar door voorlopig de kat uit de boom te kijken. Dit veranderde een jaar later, toen opinieleider Kenny Meyers zich in een emotioneel maar genuanceerd pleidooi tot EllisLab richtte met onder andere een verwijt over de mate waarin EE2.0 vernieuwingen met zich mee bracht:

> "EllisLab didn't solve problems from EE1. In fact, EE2 didn't solve anything and it's been a year. Member templates are still stuck in some weird limbo when they should be html files. MojoMotor was nice, but a bitter pill to swallow at the wrong time. We also don't know why they weren't solved. Autosaving isn't great" (Meyers, 2010)

De opsomming blijkt exemplarisch voor het type problemen waarmee de ontwikkeling van ExpressionEngine als platform te maken had. EE2.0 was een herschreven versie van de functionaliteiten van EE1.x en verwees daarmee niet alleen naar EE1.x als prototype, maar ook naar de negatieve kenmerken van deze versie, zoals de verouderde opzet van de 'member templates'. MojoMotor was een relatief simpel, maar modern CMS dat Derek Allard als een soort hobbyproject ontwikkelde: niet gehinderd door de erfenis van ExpressionEngine en daarmee volop gebruikmakend van de kracht van CodeIgniter. Meyers zag de lancering van dit product echter als een teken dat EllisLab zich niet langer enkel op de ontwikkeling van het ExpressionEngine-platform richtte en het bleek inderdaad de eerste stap van Allard naar een toekomst buiten EllisLab. En daarnaast werd er niet alleen tijd en energie gestoken in producten die weinig met het ExpressionEngine-platform van doen hadden, maar ook in functies zoals 'autosave' die er weliswaar voor nieuwe gebruikers aanlokkelijk uitzagen, maar die voor de typische ExpressionEngine-gebruiker weinig toevoegden. EllisLab leek te zijn getransformeerd van een ingenieus laboratorium dat haar uitvindingen nauwkeurig afstemde op haar loyale gebruikers, in een speelgoedwinkel die aan het meest waardevolle product allerlei ongewenste toeters en bellen verbond. Ryan Irelan, sinds jaar en dag actief als 'ExpressionEngine-verslaggever' via EEInsider.com, bracht de ontstane situatie krachtig onder woorden:

> "What is important is whether we can continue to put our own reputations behind EE 2 and sell it to our clients and customers."[17]

De bijval voor Meyers' pleidooi kende echter zijn grenzen. Voor traditionele gebruikers voegde ExpressionEngine 2.0 dan misschien weinig nieuwe functies toe, maar voor extensieontwikkelaars was de CodeIgniter-fundering wel degelijk een 'upgrade'. Het framework stelt ontwikkelaars daadwerkelijk in staat om sneller een krachtige extensie te ontwikkelen. Brian Litzinger ergert zich aan het feit dat de vernieuwingen van ExpressionEngine 2.0 slechts vanuit het perspectief van de CMS-gebruiker worden besproken:

[17] Online te vinden via http://thenerdary.net/articles/entry/a_plea_to_ellislab (laatst bezocht op 17 augustus 2011)

> "I disagree with people's statements about how 2.0 is not a step forward from 1.6 and doesn't solve anything. I think those statements are coming from people who don't actually make add-ons or know PHP"[18]

Het probleem is echter dat de mogelijke voordelen van deze integratie zich pas op langere termijn voordoen. En daarnaast hebben ontwikkelaars te maken met enkele nadelige effecten van de 'rewrite'. Het probleem dat ook extensies aangepast zouden moeten worden aan de nieuwe softwareomgeving werden door EllisLab in 2006 nog voorzien:

> "Let us not forget the modules. Since the Template parser will be receiving some Six Million Dollar Man surgery, there will be a need to rewrite the modules in order to interact with it seamlessly. The end all effect should be to make things more efficient and perhaps allow some tools that will make module writing easier than it is currently. Have no fear, third party developers. Even if your module is not rewritten for 2.0, we still intend to allow a site to continue using any current 1.x modules. Like Mac OS X, we intend to have a Classic mode to make sure no module stops working" (Burdick, 2006b)

Het ontwikkelen van een dergelijke 'classic mode' is voor Apple-ingenieurs misschien haalbaar, maar voor EllisLab bleek de complexe programmeerpraktijk weerbarstiger. Sinds 2006 was het aantal extensies enorm gegroeid. ExpressionEngine 1.x was zonder de code van derden een ander product met een stuk minder mogelijkheden en de slogan "You pretty much cannot build a website without these now" op de website van Brandon Kelly gaat daarmee ondanks de van EllisLab overgenomen schijn van overdrijving wel degelijk voor een groot deel op. De verantwoordelijkheid voor het succes van het ExpressionEngine-platform, dat inmiddels bestond uit de naast elkaar voortbestaande versies EE1.7 en EE2.0, was daarmee voor een groot deel in de handen van derden. Deze wederzijdse afhankelijkheid tussen platformaanbieder en derde partijen zien we ook in andere industrieën terug, met als bekend voorbeeld Intel. De chipmaker was altijd van zeer veel partijen afhankelijk en dit kwam met name tot uiting rondom de introductie van de 64-bit-architectuur:

> "Yet this innovation provided little benefit to users unless hundreds of companies, beginning with giants like Microsoft and Oracle, redesigned their software products to take advantage of the new 64-bit architecture" (Gawer en Cusumano, 2002: 11)

Het feit dat gebruikers ExpressionEngine 2.0 weinig vonden toevoegen, kan dan ook deels verklaard worden uit het feit dat ze vooral veel elementen moesten missen waarover EllisLab weinig controle had. Alle geliefde EE1.x-extensies waren ondanks de eerdere 'classic mode'-belofte incompatibel met CodeIgniter en moesten daarom eerst herschreven worden.

Na het debacle rondom de 'rewrite' van ExpressionEngine zelf, was het naïef om te beweren dat het herschrijven van de soms zeer complexe extensies een fluitje van een cent zou zijn. Jacob Russel, die als ontwikkelaar van extensiemarktplaats Devot-EE een

[18] Online te vinden via http://thenerdary.net/articles/entry/a_plea_to_ellislab (laatst bezocht op 17 augustus 2011)

grote rol speelde bij het bieden van zicht op de compatibiliteitsstatus van extensies, deed toch een poging:

> "You're developing a site in ExpressionEngine 2.0 and need a plugin that only has a 1.6.8 version. You could wait for the developer to update it, which could take months and might never happen. On the other hand if you're willing to get your hands dirty you can try updating it yourself. We'll tell you how, and it's not as hard as you think!" (Russel, 2010)

Voor kleine plugins gaat deze stelling wellicht op, maar Russel onderschat enerzijds de complexiteit van de meest geliefde extensies en anderzijds het feit dat inmiddels ook veel extensieontwikkelaars het 'open source' bazaarmodel achter zich hadden gelaten. Ontwikkelaars als Solspace, Lodewijk Schutte en Brandon Kelly kozen er net als EllisLab voor om het herschrijfproces achter gesloten deuren te laten plaatsvinden. Om een debacle zoals bij EllisLab te voorkomen dekte Kelly zich echter wel in tegen te hoge verwachtingen:

> "as exciting as the prospect of EE 2 is, it doesn't mean much when you rely on third party add-ons that aren't yet compatible. So I figured it's high time I revealed my roadmap for EE2 compatibility. Keep in mind that sometimes things go faster than expected, and sometimes they take longer." (Kelly, 2009)

Na twee jaar wachten op de lancering van ExpressionEngine werd het wachten dus gecontinueerd in de wachtkamers van externe ontwikkelaars. En ook deze ontwikkelaars moesten voorlopig wachten tot de vruchten van de CodeIgniter-integratie zich zouden aandienen, want voordat het feest van het schrijven van nieuwe code begon, moest eerst nog het slopende herschrijfproces van bestaande extensies afgerond worden. Niet voor het eerst in de ontwikkeling van script tot software-industrie zaten EllisLab, gebruikers en externe ontwikkelaars daarmee in hetzelfde schuitje.

5.3 Explicietere integratie

Door de jaren heen was EllisLab steeds hechter verbonden geraakt met gebruikers en externe ontwikkelaars. Gebruikers kwamen met ideeën voor nieuwe functionaliteit en verdedigden het merk ExpressionEngine met verve. Met het geïndustrialiseerde EllisLab als prototype ontwikkelden derde partijen een extensie-industrie die op een wederzijds afhankelijke manier met EllisLab verbonden raakte. EllisLab profiteerde daarmee van de voordelen van allerlei vormen van gebruikersparticipatie, maar tegelijkertijd leek het zich tegen een al te hechte verstrengeling te verzetten. Door de verregaande commercialisering werd de scheiding tussen producent en consument gekoesterd, wat tot uiting kwam in het aannemen van dominante gebruikers als werknemer. Maar in toenemende mate probeerde EllisLab zich ook minder afhankelijk te maken van derden door extensies van externe ontwikkelaars in de 'core' van ExpressionEngine op te nemen. Het was een praktijk die voorafgegaan werd door een kleine wijziging op het niveau van de programmeercode. Vanaf EE2.0 werden de bestandsmappen met daarin extensies van derde partijen niet langer tussen de mappen met door EllisLab ontworpen

modules gerangschikt, maar in een aparte map genaamd 'third_party'. Technisch veranderde hiermee weinig, maar het maakte de verplaatsing van Brandon Kelly's populaire extensie 'FieldFrame' van de map 'third_party' naar de 'core' van EE2.0 expliciet.

Enerzijds was de map 'third_party' een index van de integratie van expliciete gebruikersparticipatie, terwijl hiermee anderzijds werd bevestigd dat EllisLab moeite had met een situatie waarin een toenemend aantal extensies als essentieel werd gezien. De dubbele moraal en ongemakkelijke verhouding met de derde partijen gold sinds de introductie van EE2.0 ook voor mij en veel andere gebruikers. De module 'stand-alone-entry-form' was een onmisbaar hulpmiddel voor de ontwikkeling van al mijn EE1.x-websites en het feit dat EllisLab een jaar na de lancering van EE2.0 deze functie nog steeds niet aan het nieuwe product toegevoegd had, belette mij te upgraden naar EE2.0. De commerciële extensie 'Safecracker' bleek dit gemis op te lossen en schafte ik daarom met plezier voor 99 dollar aan, maar niet veel later was ik minstens zo gelukkig met het feit dat EllisLab deze extensie overnam en hem voortaan gratis via de 'core' beschikbaar stelde. EllisLab en gebruikers houden er dus een dubbele moraal ten aanzien van gebruikersparticipatie op na: eerst wordt de integratie van participatie in de structuur van het CMS als platform toegejuicht, om vervolgens te worden afgedankt als zijnde een ongewenste extra afhankelijkheidsrelatie. Het zelf ontwikkelen of kopen van extensies die anders in het beheer van externe ontwikkelaars zou zijn geweest, kan daarom opgevat worden als een alternatief of in termen van Latour een 'anti-program' jegens gebruikersparticipatie, dat we ook rondom andere platforms terugzien:

> "alternatives include making essential complementary products yourself (like Microsoft), or buying promising third-party complementors (like Cisco)"
> (Gawer en Cusumano, 2002: 12)

Waar enkele gebruikers de voordelen inzien van de verplaatsing van 'third_party' naar 'core', waarschuwt Brian Litzinger voor de gevaren die op de loer liggen:

> "stop complaining about having to rely on 3rd party add-ons such as Playa and Matrix. They should _not_ be part of the core. If add-ons like that are incorporated into the core, it just becomes a bloated behemoth of a CMS, and more EllisLab needs to maintain, thus slower development cycles. The beauty of EE is it's add-ons are solid, and generally so integrated they don't really feel like add-ons (unlike Drupal and WP plugins). There are a few black sheep that insist on using their own UI, and for that I don't use them"[19]

Ontwikkelaar Litzinger ziet ExpressionEngine 2.0 vooral als platform, voor zowel extensies als webapplicaties, dat door middel van de keuze van de juiste bouwstenen in de vorm naar keuze gegoten kan worden. Het is de richting waarin ExpressionEngine zich vanaf 2001 ontwikkeld heeft, met als duurzaam resultaat dat ExpressionEngine's meest kenmerkende 'templatetag' sinds EE2.0 niet langer de tekst

[19] Online te vinden via http://thenerdary.net/articles/entry/a_plea_to_ellislab (laatst bezocht op 17 augustus 2011)

‘{exp:weblog:entries}’ maar ‘{exp:channel:entries}’ draagt. Zolang EllisLab en een aanzienlijk deel van de gebruikers ExpressionEngine niet puur als platform ontwikkelen, omschrijven en gebruiken, blijven we de erfenissen van deze praktijken, en de erfenissen van de geschiedenis van ExpressionEngine als weblog en CMS waarmee ‘out-of-the-box’ standaard websites gemaakt konden worden, in het systeem terugzien.

6. Conclusie: Relationships Made Durable

Na een avontuurlijke reis door het veranderlijke, heterogene landschap rondom ExpressionEngine zijn we teruggekeerd bij de vraag waarmee dit onderzoek begon: *hoe heeft de dynamiek tussen EllisLab, het softwareproduct ExpressionEngine en verschillende derde partijen zich gedurende de afgelopen tien jaar ontwikkeld?*

De originators van ExpressionEngine (eerst Rick Ellis en later de medewerkers van EllisLab) en het heterogene en veranderende gezelschap individuen dat samen de 'EE-community' vormt, zijn gedurende het afgelopen decennium verstrikt geraakt in een gecompliceerd web van wederzijdse verwachtingen.

Veel participanten brengen een geschiedenis of andere externe associaties met zich mee, zoals opvattingen over softwareontwikkeling, kennis van een softwaretaal, klanten met specifieke wensen, een bestaande website draaiende op Joomla, Drupal of Wordpress, een visie op ondernemerschap en/of een gezin dat droomt van een dure wereldreis. Deze 'extensies' worden in veel gevallen niet gedeeld met de rest van de community, maar kunnen wel degelijk andere participanten, waaronder het softwareproduct ExpressionEngine, beïnvloeden. Dit onderzoek bevestigt de stelling van onder andere Schäfer (2008) en Mackenzie (2006) dat een actor-netwerk-analyse in termen van associaties en transformaties deze vluchtige en duurzame invloeden op de ontwikkeling van hedendaagse software aan de oppervlakte kan brengen.

Hoofdstuk drie liet zien welke actoren een rol van betekenis speelden bij de ontwikkeling van de voorvader van het huidige ExpressionEngine: pMachine. De specifieke situatie waarin 'originator' Rick Ellis zich in 2001 bevond kan gerelateerd worden aan de specifieke technische specificaties van pMachine, de specifieke 'affordances' van pMachine en de specifieke manier waarop pMachine gedistribueerd werd. Vervolgens bleek dat de richting van de ontwikkeling van pMachine in eerste instantie geassocieerd kan worden aan de specifieke manier waarop pMachine werd gebruikt en becommentarieerd door de groep 'recipients' die mede dankzij persaandacht op het product afkwam. Ellis construeerde aan de hand van de kenmerken van deze onvermoede 'appropriatie' en zijn eigen ambities een nieuwe toekomstvisie die tot uiting kwam in het bedrijf EllisLab en het commerciële softwareproduct ExpressionEngine. Het was een visie die de kiem vormt van een ontluikende complexe relatie tussen EllisLab, ExpressionEngine en verschillende typen gebruikers.

Hoofdstuk vier maakte duidelijk dat dit nieuwe product de erfenis van pMachine en hoofdontwikkelaar Rick Ellis met zich meedroeg, maar dat de verdere ontwikkeling tevens hevig werd beïnvloed door diverse nieuwe actoren. De professionaliseringsslag van EllisLab en inrichting van een CMS gericht op professionele bloggers en ontwerpers bleek aantrekkelijk voor zelfstandige webprofessionals en webontwerpbureau's. Veel van deze gebruikers identificeerden zich net als EllisLab zowel met de beloftes van de opensourcegedachte en amateurcultuur als met commerciële businessmodellen en de daarmee gepaard gaande zakelijke cultuur. Zij droegen deze 'perfecte combinatie' dan ook graag over: aan elkaar, aan klanten en aan iedereen die overtuigd was van de superioriteit van Wordpress, Joomla en Drupal. Net als deze concurrenten was

ExpressionEngine vanaf versie 1.5 ontworpen om door middel van 'plugins', 'modules' en 'extensions' uitgebreid te worden. Met de ontwikkeling van EllisLab/ pMachine/ExpressionEngine als 'prototype' werd deze vorm van 'explicit participation' steeds professioneler, met als resultaat dat deze 'EEndustrie' wat betreft kosten van softwareproducten, support, overige diensten, marketinguitingen en bedrijfsvormen steeds sterker op het 'core'-product en 'moederbedrijf' begon te lijken. De door Schäfer (2008) omschreven definitie van gebruikersparticipatie als een 'extension of the cultural industries' kunnen we ons met betrekking tot expliciete participatie rondom ExpressionEngine daarom in opvallend letterlijke zin toe-eigenen.

De professionalisering van de verschillende participanten binnen het 'EEcosysteem' lijkt in eerste instantie vrijwel alle betrokken partijen ten goede te komen. In hoofdstuk vijf bleek echter dat het ontstaan van 'platforms of platforms of platforms' en de hechte verstrengeling van hoofdproduct en derden de verdere ontwikkeling van het product kan tegenwerken. Waar ExpressionEngine 1.0 erfenissen van pMachine met zich meebracht, heeft elke volgende software-update te maken met de erfenissen van voorgaande versies en met name ook de erfenissen van de (software-)extensies van die eerdere versies. Met een videopreview van ExpressionEngine 2.0 beloofde EllisLab dat het deze problemen te lijf kon. De onmogelijkheid van het verwijderen of 'upgraden' van de erfenis van vier jaar ExpressionEngine 1.x zorgde echter voor interne strubbelingen binnen EllisLab en teleurgestelde reacties van gebruikers toen ExpressionEngine 2.0 na lang wachten in de vorm van een 'beta' gelanceerd werd. Formeel gezien was ExpressionEngine 2.0 een veelbelovende extensie van EllisLab's 'open source' PHP-framework CodeIgniter, maar inzicht in het veelbesproken pleidooi van 'lead user' Kenny Meyers maakt duidelijk dat discussie op basis van dergelijke formele systemen altijd gepaard moet gaan met het openen van de talloze 'black boxes' rondom softwareontwikkeling, waarbij zowel de meningen, memoires en andere publicaties van betrokkenen als het aannemen van de rol participerende observator van essentieel belang zijn.

Deze actor-netwerk-analyse van de ontwikkeling van ExpressionEngine bevestigt hiermee een groot deel van de conclusies van onder andere Schäfer (2008) en Van der Graaf (2009). De conclusies die Van der Graaf ontleidt aan het softwareplatform Second Life komen in grote mate overeen met mijn bevindingen:

> "such a dynamic of 'give and take' among constellations of contributing developers demonstrates an interdependent relationship that suggests a consolidated life cycle underlying product development that is simultaneously structured and emergent, top-down and bottom-up, centralized and dispersed, commercial and non-commercial." (Van der Graaf, 2009: 214)

Wat het onderzoek toevoegt ten opzichte van het onderzoek van Van der Graaf is dat het niet een gestabiliseerd object onderzoekt, maar een ontwikkeling die het stabiliseren van het onderzoeksobject zichtbaar maakt en daarmee op Latouriaanse wijze verklaart. De focus op de ontwikkeling van één softwareproduct gedurende een periode van tien jaar maakt het onderzoek tevens tot een waardevolle extensie van

Schäfer (2008). Het onderzoek brengt nauwkeurig in kaart hoe verschijningsvormen van gebruikersparticipatie rondom een zich ontwikkelend softwareproduct steeds weer nieuwe vormen aannemen, waarbij Schäfer's 'black boxes' 'expliciete participatie' en 'integratie van participatie' vooral waarde toevoegen wanneer ze geopend worden.

De ExpressionEngine-ecologie heeft zich in dit onderzoek gepresenteerd als een object dat zich met veel liefde liet onderzoeken. De liefdesrelatie tussen EllisLab, gebruikers en derde partijen weet de crisis rond de introductie van EE2.0 voorlopig te doorstaan, waardoor wederom een nieuwe fase aanbreekt in de ontwikkeling van deze dynamische, maar duurzame software-ecologie. Waar dit onderzoek de ontwikkeling van de ExpressionEngine-geschiedenis voorafgaande aan directeur Camacho's statement "We love you guys, we really do" beschreef, zou vervolgonderzoek juist de ontwikkelingen vanaf dit statement in kaart kunnen brengen. Als participerende observator blijf ik nauw betrokken bij de spannende tijd die ExpressionEngine tegemoet gaat, maar met het naderen van het einde van mijn master Nieuwe Media en Digitale Cultuur wordt mijn rol binnen de mediawetenschappen (tijdelijk) beperkt. Nu ik heb laten zien dat Mackenzie's begrippen 'code as index', 'originators', 'recipients' en 'prototypes' zich uitstekend lenen voor een uitgebreide analyse volgens de principes van de actor-network theory, hoop ik echter dat mijn beoogde vervolgonderzoek naar ExpressionEngine deels door andere onderzoekers binnen 'software studies' en 'platform studies' opgepakt wordt. Wellicht groeit deze 2500 regels lange 'open source' extensie van gevestigd participatieonderzoek dan ooit zelf uit tot een complexe kennisindustrie.

7.

Bibliografie

Allard, Derek. 2008. ExpressionEngine 2.0: Fully CodeIgnited!. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/expressionengine_20_fully_codeignited/.

Bijker, Wiebe; en John Law, eds. 1992. Shaping Technology/Building Society: Studies in Sociothechnological Change. MIT Press. Cambridge, MA.

Bogost, Ian; en Nick Montfort. 2009. Platform Studies: Frequently Questioned Answers. Laatst bezocht op 10 augustus 2011. http://www.escholarship.org/uc/item/01r0k9br.

Boomen, Marianne van den, en Mirko Tobias Schäfer. 2005. Will the Revolution be Open Sourced? How open source travels through society. In: Wynants, Marleen; Cornelis, Jan (Eds.): *How Open Is the Future? Economic, Social and Cultural Scenarios inspired by Free & Open Source Software*. VUB Press. Brussel. pp. 31-68.

Burdick, Paul. 2006a. The Next Big Thing. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/the_next_big_thing/.

— —. 2006b. Developers, Developers, Developers. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/developers_developers_developers/.

Camacho, Leslie. 2009a. Not 2.0's Gorilla. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/not_2.0s_gorilla/.

— —, 2009b. You Spoke, We Listened: Important Release Updates. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/important_release_updates/.

— —. 2010. I Hear You. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/i_hear_you/.

Copier, Marinka. 2007. *Beyond the magic circle: A network perspective on role-play in online games*. Dissertatie. Utrecht University. Utrecht.

Ellis, Rick. 2006a. Those Crazy VCs. Laatst bezocht op 17 augustus 2011. http://www.ellislab.com/index.php/those_crazy_vcs/.

— —. 2006b. ExpressionEngine is Dead!. Laatst bezocht op 17 augustus 2011. http://www.ellislab.com/index.php/expressionengine_is_dead/.

— —, 2007a. Building an Ecosystem. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/building_an_ecosystem/.

— —, 2007b. ExpressionEngine Turned Three. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/expressionengine_turned_three/.

— —, 2008. ExpressionEngine 2.0 Delayed. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/expressionengine_20_delayed/.

Fuller, Matthew. 2003. *Behind the blip: Essays on the culture of software*. Autonomedia. Brooklyn.

— —, ed. 2008. *Software Studies: A Lexicon*. MIT Press. Cambridge, MA.

Galloway, Alex. 2005. *Protocol: How Control Exists After Decentralization*. MIT Press. Cambridge, MA.

Gawer, Annabelle; en Michael Cusumano. 2002. *Platform Leadership*: H*ow Intel Microsoft and Cisco Drive Industry Innovation*. Harvard Business School Press. New York.

Graaf, Shenja van der. 2009. *Designing for Mod Development: User creativity as product development strategy on the firm-hosted 3D software platform*. Unpublished Ph.D. Londen School of Economics and Political Science. London.

Hippel, Eric von. 2005. *Democratizing Innovation*. MIT Press. Cambridge, MA.

Howe, Jeff. 2008. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. Crown Business. London.

Jones, Derek. 2006. It's 3 in the A.M. and You're Watching Perspectives. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/its_3_in_the_am_and_youre_watching_perspectives/.

— —. 2008a. I See Dead Server-Side Scripting Languages. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/i_see_dead_server_side_scripting_languages/.

— —. 2008b. What's Taking So Long?!. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/whats_taking_so_long/.

— —. 2008c. A Few Words on EE vs. WordPress, Joomla, et al. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/archived_forums/viewthread/99989/P18/.

— —. 2010. Acknowledge, Identify, React. Laatst bezocht op 17 augustus 2011. http://expressionengine.com/blog/entry/acknowledge_identify_react/.

Kelly, Brandon. 2010a. Custom Fields and the Death of the Field Group. Laatst bezocht op 17 augustus 2011. http://brandon-kelly.com/blog/custom-fields.

— —. 2010b. Making P&T: Juniper. Laatst bezocht op 17 augustus 2011. http://pixelandtonic.com/blog/making-pt-juniper.

Latour, Bruno. 1987. *Science in Action: How to Follow Scientists and Engineers Through Society*. Harvard University Press. Cambridge, MA.

— —. 1991. Technology is Society Made Durable. In: Law, John: *A Sociology of Monsters: Essays on Power Technology and Domination.* Routledge. Londen. pp. 103-131.

— —. 1993. *We Have Never Been Modern*. Harvard University Press. Cambridge, MA.

— —. 1999. *Pandora's Hope: Essays on the Reality of Science Studies*. Harvard University Press. Cambridge, MA.

— —. 2005b. *Reassembling the Social: an Introduction to Actor-Network-Theory*. Oxford University Press. Oxford.

Law, John. 1991. *A Sociology of Monsters: Essays on Power, Technology and Domination.* Routledge. London.

Mackenzie, Adrian. 2006. *Cutting Code: Software and Sociality*. Peter Lang Publishing. New York.

Manovich, Lev. 2001. *The Language of New Media*. MIT Press. Cambridge, MA.

— —. 2008. *Software Takes Command*. Niet gepubliceerd boek. Laatst bezocht op 10 augustus 2011. http://lab.softwarestudies.com/2008/11/softbook.html.

Meyers, Kenny. 2010. A Plea to EllisLab. Laatst bezocht op 17 augustus 2011. http://thenerdary.net/articles/entry/a_plea_to_ellislab.

— —. 2011. A Thanks for EllisLab. Laatst bezocht op 17 augustus 2011. http://thenerdary.net/articles/entry/a_thanks_for_ellislab.

Pieters, Veerle, et al. 2010. The Full Story of my ExpressionEngine 2 Design Involvement. Laatst bezocht op 17 augustus 2011. http://veerle.duoh.com/design/article/the_full_story_of_my_expressionengine_2_design_involvement.

Raymond, Eric. 1998. The Cathedral and the Bazaar. *First Monday.* Vol 3. No 3. http://www.firstmonday.org/issues/issue3_3/raymond/.

Rieder, Bernhard, and Mirko Tobias Schäfer. 2008. Beyond Engineering: Software Design as Bridge over the Culture/Technology Dichotomy. In: *Philosophy and Design: From Engineering to Architecture*. Pieter E. Vermaas et al. eds. Springer: Dordrecht. pp 152-164.

Russel, Jacob. 2010. Adventures in Updating an ExpressionEngine Plugin from 1.6.8 to 2.0. Laatst bezocht op 17 augustus 2011. http://devot-ee.com/articles/item/adventures-in-updating-an-expressionengine-plugin-from-1.6.8-to-2.0.

Schäfer, Mirko Tobias. 2008. *Bastard Culture! User Participation and the Extension of Cultural Industries*. Dissertatie. Utrecht University. Utrecht.

Schutte, Lodewijk. 2011. Launch: Go To Low. Laatst bezocht op 17 augustus 2011. http://gotolow.com/blog/launch.

Tapscott, Don, Anthony D. Williams. 2006. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio. New York.

Vries, Imar de. 2008. *Tantalizingly close: An Archeology of Communication Desires in Discourses of Mobile Wireless Media*. Dissertatie. Universiteit Utrecht. Utrecht.