dOPE
de Online Programming Editor

Kevin Valk Leon van der Veen Ben Vaassen

Juli 2010

Contents

Abstract

Als studenten informatica hebben wij logischerwijs programmeervakken. Deze werden meestal uitgevoerd in groepjes van twee. Een van onze ergernissen, was het ontbreken van een gezamenlijke werkomgeving die ons toestond om samen aan een programmeeropdracht te werken.
Hierop hebben wij dOPE bedacht, een online programming editor waarin je online aan een project kan werken en online kan slaan. Hierdoor vergeet je nooit een opdracht op te slaan en kan je tegelijkertijd werken op verschillende pc's aan hetzelfde project.
We zijn uitgegaan van onze eigen ervaring en ergernissen. Op basis hiervan hebben wij de elementen die centraal staan in dOPE bepaald (op willekeurige volgorde):

1. Projectwerk.
2. Online Programmeren.

3. Modulair.
4. Documentatie.
5. Gebruiksvriendelijk

We hebben gebruik gemaakt van javascript en PHP om de webapplicatie te schrijven. Het merendeel hebben we zelf geschreven, hoewel we wel gebruik gemaakt hebben van bestaande libraries en/of (onderdelen van) editors. Als framework hebben we Kohanna gebruikt en als editor MirrorCode.
In de praktijk bleek ons project niet volledig haalbaar binnen de gegeven tijd, maar omdat dOPE volledig modulair is, was het voor ons mogelijk om wel de basis af te krijgen. Het resultaat is te zien op de dOPE site **(klik)**. Updates zijn mogelijk in de vorm van modules.

1 Theoretisch Kader

Om te beginnen is er een editor nodig. Een editor is een programma waarmee je bestanden kunt bewerken.

> The most important characteristic of an editor is its convenience for the user. [2]

Ook voor ons was dit het belangrijkste punt waar we rekening mee moesten houden. Gebruiksvriendelijkheid betekent voor ons in de editor:

1. Autocompletion
2. Tab/Spatie inspringen
3. Color highlighting van syntax

In de komende subsecties gaan we het eerst hebben over een online programming editor. Ten tweede gaan we hebben over verschillende (bestaande) ontwerpen. Als derde gaan we de opbouw van ons dOPE systeem bespreken en als vierde en laatste leggen we de problemen waar we tegen aan kunnen lopen uit.

1.1 Een online programming editor

> In discussing editors firm opinions abound: everyone, from greenhorn to old hand, knows exactly what the best and worst features of given editors are and just how new editors ought to be designed. The only problem is the striking lack of consensus. Nor are there universally acceptable means of determining who is right: the distinction between conventional dogma and scientific fact is often blurred.[3]

Uit bovenstaand citaat blijkt dat er geen duidelijke standaard is voor editors. Editors worden geschreven naar de smaak van de ontwikkelaar. Dit klinkt ook logisch, aangezien de ontwikkelaar denkt dat hij bestaande editors kan verbeteren of een gehele nieuwe editor schrijven, die beter is dan de bestaande

editors. Op deze manier zijn ook wij begonnen. We hebben met ons drieen gekeken naar bestaande editors en aan de hand daarvan besloten wat centraal staat voor onze dOPE.

1. Projectwerk.
2. Online Programmeren.
3. Modulair.
4. Documentatie.
5. Gebruiksvriendelijk

Nu we hebben uitgelegd wat een editor is en wat we in de editor willen verwerken, kunnen we een stap verder gaan. Er bestaan al enkele online editors, waar je tekst tegelijkertijd kan bewerken en online kan opslaan. Er bestaan ook al enkele online programming editors, die een bepaalde taal ondersteunen, zoals Java of C++. Wij willen echter dat het mogelijk wordt om elke programmeertaal toe te voegen, door het schrijven van een module. Wij zullen zelf slechts enkele talen toevoegen, vanwege tijdgebrek, maar het is dus mogelijk om je elke taal toe te voegen die je wilt.

1.2 Verschillende (bestaande) ontwerpen

Het ligt voor de hand dat wij niet de eersten zijn die op het idee van een online programming editor zijn gekomen. Er zijn al enkele goede online editors op internet te vinden en die ook open source zijn. Bijvoorbeeld: Bespin **(klik)**, Jedit **(klik)** en JPedit **(klik)**. We zullen elk van deze online programming editors bekijken aan de hand van een screenshot. We beschrijven wat we zien en wat wij, als gebruikers, ervan vinden. Aangezien wijzelf de eindgebruiker zijn, dient de editor aan de eisen te voldoen die wij stellen.

1. Bespin 1
   De eenvoud in dit design springt naar voren. Het ziet er overzichtelijk uit en er zijn weinig extra knoppen. De knoppen die er wel zijn, geven duidelijk aan waar ze voor bedoeld zijn. Er zijn regelnummers aangegeven en je kunt de geschiedenis bekijken. Er is gebruik gemaakt van kleuren, wat bevorderlijk is voor het lezen van de code. Er is geen overzicht van de projecten waar je op dit moment aan werkt.

2. Jedit 2
   Waar begin je in deze editor? Het is niet echt duidelijk wat waarbij hoort. Er is erg veel gebruik gemaakt van veel tabbladen en extra vensters in vensters. Nadat je het geheel beter hebt bekeken, hebben de knoppen een duidelijke betekenis. Er is een overzicht van je projecten en wat er in elk project zit. De tabbladen zijn erg handig en houden het overzicht. Hier is echter gebruik gemaakt van zowel horizontale als verticale tabbladen. Het lijkt ons handiger als dit of allemaal horizontale, of allemaal verticale

tabbladen zijn, om de regelmaat en het overzicht te behouden. Er is gebruik gemaakt van kleuren om gemakkelijker de code te kunnen lezen. Ook zijn er regelnummers aangegeven.

3. JPedit 3
   Er is een duidelijk overzicht. Er zijn vier verschillende onderdelen die elk een doel hebben. De knoppen zijn duidelijk en overzichtelijk. Er is gebruik gemaakt van kleuren om het lezen van de code makkelijker te maken. Er zijn geen regelnummers aangegeven. Het lijkt ons handig om zelf te bepalen waar je welk onderdeel kan neerzetten en dat je elk onderdeel kan verbergen of vergroten.

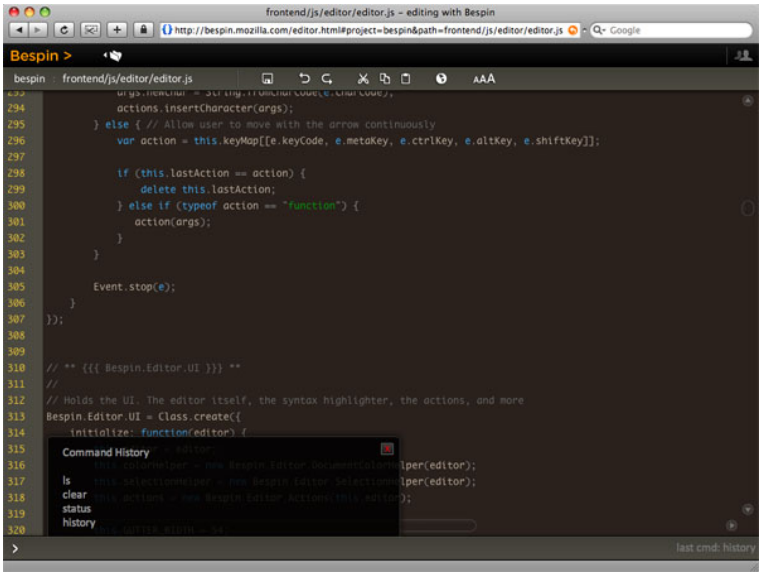

Figure 1: Bespin

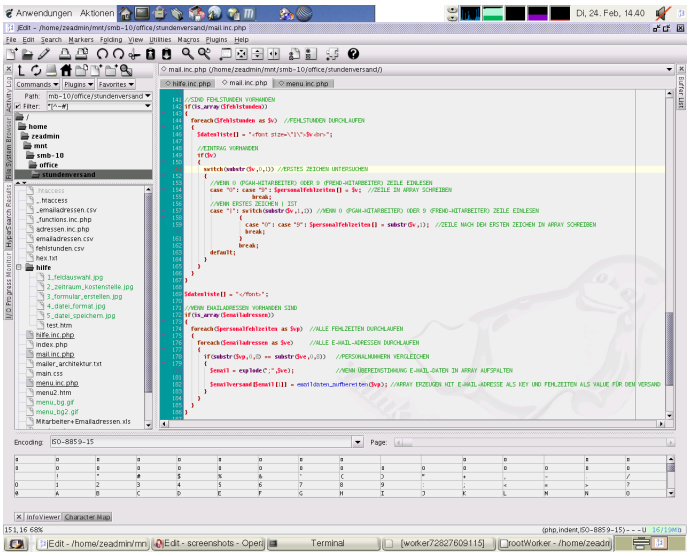

Figure 2: Jedit

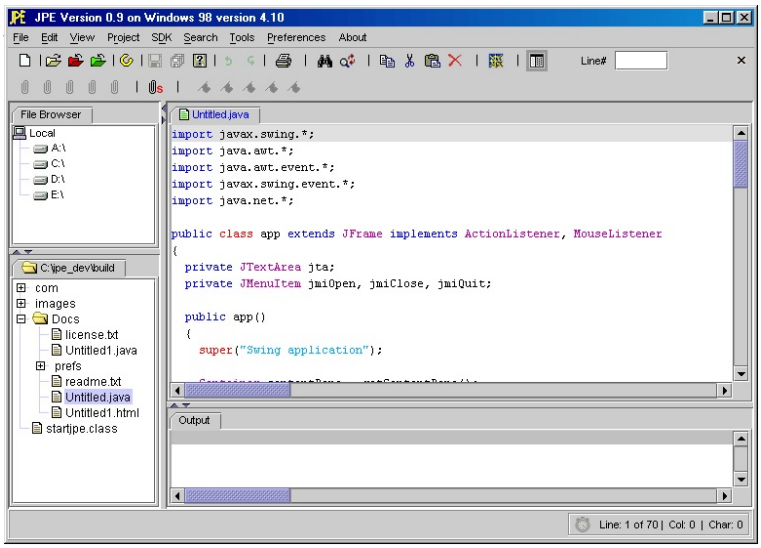

Figure 3: JPedit

1.3 Opbouw dOPE

We hebben besloten om dOPE op te bouwen uit verschillende modules, zodat het makkelijk uitbreidbaar is. We werken volgens het MVC principe:

> This three-way division of an application entails separating (1) the parts that represent the model of the underlying application domain from (2) the way the model is presented to the user and from (3) the way the user interacts with it.[4]

Dit is een belangrijk principe waar we gebruik van maken. Door dit principe is het mogelijk om op een modulaire manier, dOPE uit te breiden. Om dOPE te realiseren hebben we een aantal elementen uitgewerkt die we willen opnemen in ons systeem:

1. Framework

   > Object-oriented frameworks are widely used to implement reusable designs that can be completed to build custom applications. An object-oriented framework consists of core classes that implement core functionality and application programming interface (API) classes that can be extended or instantiated in the application code.[1]

   Het framework vormt dus de basis voor ons dOPE systeem en geeft ons de mogelijkheid om een standaard functionaliteit toe te voegen.

2. Projectmanager
   De projectmanager beheerst de verschillende projecten waar jij op dit moment mee bezig bent. Je hebt een overzicht van alle projecten waar jij op dit moment voor bent aangemeld. Je kunt nieuwe projecten aanmaken en andere projecten toevoegen aan een huidig project. Als een project wordt toegevoegd aan je huidige project, dan zal dit een hard copy worden van de source code. Maar er wordt ook gebruik gemaakt van slim sync die het mogelijk maakt om deze harde copy up-to-date te houden.

3. Editor
   In de editor kun je programmeren. Je schrijft hierin je programma. We willen dat dit zo gebruiksvriendelijk mogelijk gebeurd, dus willen we zorgen voor syntax highlighting, overzichtelijkheid (door middel van tabbladen), autocompletion en collaborative real time editten.

4. Rechten
   Om ons dOPE systeem goed te laten werken, moet je bepaalde rechten kunnen verlenen aan andere gebruikers. Je kan op elk gewenst moment bepalen of je project een public source of een closed source project is. Een open source project laat de code die je hebt gemaakt aan iedereen zien en als je een uitnodiging krijgt van de eigenaar van het project, kun

je meewerken. Als je gaat werken aan een closed source project, kunnen andere gebruikers je code niet zien.

5. Settings per project
   Per project moet je bepaalde instellingen kunnen aanpassen. Je moet kunnen selecteren in welke taal je wilt programmeren, maar bijvoorbeeld ook welke compiler en layout je wilt gebruiken.

6. Documentatie
   In de documentatie kun je bijvoorbeeld de programmeeropdracht opslaan, zodat je deze altijd makkelijk bij de hand hebt. Maar je moet er ook kunnen omschrijven wat je project inhoudt en eventueel in UML een project kunnen opbouwen.

1.4 Problemen

Het grootste probleem bij het schrijven van je eigen programma is dat je nooit weet of je programma wel binnen de tijd die gegeven is, afgemaakt kan worden. Afgezien van de tijdsdruk, kunnen er nog meer problemen tevoorschijn komen. We gaan er nu van uit dat als we in tijdsdruk komen, we onderdelen van andere programma's of systemen kunnen gebruiken en dat deze dan ook meteen werken. Dit hoeft niet perse zo te zijn en dan zijn we alsnog genoodzaakt om alles zelf te schrijven. Afhankelijk van hoeveel tijd er nog over is, zou dit wel op te lossen zijn.
Verder zou de code die we schrijven heel goed geschikt kunnen zijn voor een bepaalde webbrowser, maar dat hoeft niet perse te gelden voor alle webbrowsers. Het oplossen van dit probleem zou erg lang kunnen duren en we zouden er dan ook voor kunnen kiezen om dit achterwege te laten en slechts een webbrowser te ondersteunen (en natuurlijk de andere webbrowsers waar het toevallig ook goed werkt).

Uibreiden misschien?

2 Methode

We willen een eigen online programming editor schrijven. We kunnen vanuit niets beginnen en alles zelf schrijven. Maar het ligt voor de hand dat we het wiel niet opnieuw hoeven uit te vinden. We hebben gekeken naar andere programma's en de goede elementen eruit gehaald en in ons eigen systeem gestopt. Hierdoor combineren we het beste van elk afzonderlijk systeem in een dOPE systeem. We gaan eerst kijken naar de taal waarin we gaan programmeren. Hierna bespreken we het Framework. Dit wordt gevolgd door de Editor en wordt afgesloten door de documentatie.

2.1 Programmeertaal

Als eerste moesten we besluiten in welke taal we gaan programmeren. Doordat we in de Pilot begonnen waren met vooronderzoek naar de editor, hebben we bij de editor veel meer mogelijkheden gevonden dan wanneer we de programmeertaal eerst hadden bepaald.
Om te besluiten in welke taal we gaan programmeren, hebben we gekeken naar verschillende talen en de mogelijkheden die elke taal bood. Ook was het van belang dat we ervaring moesten hebben met de taal, zodat we geen tijd hoefden te besteden aan het leren van een hele nieuwe programmeertaal. We zijn begonnen met de talen waarin we ervaring hadden en op basis hiervan hebben we een lijstje gemaakt met geschikte talen:

- PHP / HTML (4/5) / Javascript (Ajax)
- PHP / JAVA (Applet)
- PHP / Flash

Om duidelijk te krijgen welke taal welke voor- en nadelen heeft maken we een overzicht:

	Javascript + HTML 4	Javascript + HTML 5	Java	Flash (Actionscript)
Platform onafhankelijk [1]	Ja	Ja	Ja [2]	Ja [3]
Traag [4]	Nee	Nee	Ja / Nee	Ja
Runtime script [5]	Ja	Ja	Nee	Nee

1. OS / Browser onafhankelijk
2. Mits JRE aanwezig
3. Mits Flash-plugin aanwezig
4. Gevoel van haperingen
5. Heeft geen compile tijd

De data is gebaseerd op **(klik)**. Na het bekijken van de data en de ervaring die we hebben met elke taal, is de keuze voor de programmeertaal op Javascript gevallen. De volgende redenen waren doorslaggevend bij het bepalen van de taal:

- Javascript werkt in iedere browser (geen noodzaak tot installeren van plug-ins).
- Javascript is relatief snel (Java is sneller, maar het laden van een Java Applet duurt lang omdat bestanden groot zijn en het initialiseren van Java duurt lang).

- In Javascript is het makkelijkst te ontwikkelen (geen compiler nodig, goede support door editors).
- Geeft goede mogelijkheid tot HTML lay-out features.
- De laad tijd
- De responsie tussen de gebruiker en de applicatie
- Ervaring van Leon en Kevin met Javascript

2.2 Framework

Hierna zijn we aan de slag gegaan met het framework. In de pilot waren we van plan om deze helemaal zelf te schrijven. Vanwege te weinig tijd, zijn we hiervan afgestapt en hebben we besloten om een bestaand PHP framework te gebruiken. Een (goed) framework bied een grote standaard functionaliteit zoals database query builders, sessions, etc.
De keuze is nu gevallen op het Kohana: The Swift PHP Framework (v3.x). Kohana (v3.x) is een light weight maar krachtig PHP framework. De keuzen is hier op gevallen door Leon, die er al ervaring mee had. Een groot voordeel van Kohana is de structuur waarin applicaties gemaakt worden: het 'cascading filesystem' en het HMVC pattern (Hierarchical-Model-View-Controller pattern, verbeterd MVC principe). Kohana is ook een goed gemplementeerd modulair framework. Zo kan je makkelijk extra functionaliteit toevoegen die automatisch door het hele framework gebruikt kan worden. Dit kan handig toegepast worden met plug-ins voor dOPE (bijv. een extra programmeertaal voor de editor).
Het HMVC pattern word nog niet zo veel toegepast. Kohana is een van de eerste, bekendere, PHP frameworks dit HMVC toe past. Voor meer informatie over HMVC, volg deze link: **(klik)**.

2.3 Editor

We hebben ook onderzoek gedaan naar verschillende bestaande editors, om te kijken of wij er een deel van kunnen gebruiken of zelfs in het geheel kunnen overnemen. We hebben een lijst gemaakt van de editors die ons hiervoor geschikt leken. We hebben ze gerangschikt op de programmeertaal, zodat er een duidelijk overzicht is.

- ***Java***

    1. **Jedit (klik)**
       jEdit is a mature programmer's text editor with hundreds (counting the time developing plugins) of person-years of development behind it. To download, install, and set up jEdit as quickly and painlessly as possible, go to the Quick Start page. While jEdit beats many expensive development tools for features and ease of use, it is released

as free software with full source code, provided under the terms of the GPL 2.0. The jEdit core, together with a large collection of plugins is maintained by a world-wide developer team.
Some of jEdit's features include:

- Written in Java, so it runs on Mac OS X, OS/2, Unix, VMS and Windows.
- Built-in macro language; extensible plugin architecture. Dozens of macros and plugins available.
- Plugins can be downloaded and installed from within jEdit using the "plugin manager" feature.
- Auto indent, and syntax highlighting for more than 130 languages.
- Supports a large number of character encodings including UTF8 and Unicode.
- Folding for selectively hiding regions of text.
- Word wrap.
- Highly configurable and customizable.
- Every other feature, both basic and advanced, you would expect to find in a text editor. See the Features page for a full list.

2. **JPedit (klik)**
   JPE is a pure Java text editor written for my final year project at Bolton Institute of Higher Education. It is freeware and open source. The original editor was quite basic but I've re-written a more advanced version with more features which is now available for download. The editor uses the syntax highlighting package from an early version of JEdit which is freely available for non-commercial use. The software was in a constant state of development up until recently, but work commitments have slowed things down a bit. I hope to get back to adding more features and bug fixing in the not too distant future. Although the editor was only written for a college project and not for commercial use, it is usable and stable. Some aspects, such as searching and replacing are still a bit buggy as is the compiler preferences screen and as such, the version remains at 0.9. As with most freeware and open source software the editor comes with no guarantee and you use it at your own risk(see bottom of page).

- ***Javascript***

1. **jsvi (klik)**
   If you saw Adam's recent Hive Five roundup of text editors, you might have noticed that Vim, a child of Unix/Linux favorite Vi, still carries a lot of favor among coders and back-to-basics text workers. Now you can try out Vi and all its shortcut/macro goodness online with jsvi, a JavaScript-written clone of the basic Vi interface. It's

obviously focused on code, carrying substitutions and spell checking for the most common languages, but it's a fun place to try out coding for newcomers, or for programmers to do a little quick hacking when they're away from their systems.

2. **SyntaxHighlighter (klik)**
   SyntaxHighlighter is a fully functional self-contained code syntax highlighter developed in JavaScript. To get an idea of what SyntaxHighlighter is capable of, have a look at the demo page. The project was started in 2004 and since then has gained a lot of acceptance. Version 2.0 is the new page in history of the project representing a near complete rewrite, clean up, optimization, standard compliance and new features.
3. **JSHighlighter (klik)**
   JSHighlighter is a JavaScript dedicated Syntax Highlighter. Maybe one day I'll modify JSHighLighter to parse different languages :-) JSHighlighter requires JSL and then is compatible with all JavaScript 1.2 browser, starting from IE4.
   Which are JSHighlighter features:
   - Really fast basic highlight that should be used with some other languages too (ActionScript and maybe others).
   - Complete full highlight (not fast as basic convertion with old CPUs).
   - Compatible with all JS 1.2 or greater browser.
   - Unobtrusive way to highlight javascript source code.
4. **CodeMirror (klik)**
   CodeMirror is a JavaScript library that can be used to create a relatively pleasant editor interface for code-like content computer programs, HTML markup, and similar. If a parser has been written for the language you are editing (see below for a list of supported languages), the code will be coloured, and the editor will help you with indentation.

- ***Flash***

1. **Flash Text Formatter (klik)**
   Flash Text Formatter (FTF) can format the text according to keyword definition list stored in external XML file. Its primary purpose is code syntax highlighting. At this moment it supports:
   - ActionScript 2
   - PHP
   - JavaScript
   - Python

Writing keyword definition XML is easy, so expect this list to grow fast.

Nadat we de programmeertaal hadden bepaald, was het duidelijk dat Flash en Java editors afvielen. We hadden toen nog de keuze tussen jsvi, SyntaxHighlighter, JSHighlighter en CodeMirror. Onze voorkeur ging uit naar CodeMirror vanwege 3 redenen:

- Hij is gemakkelijk uit te breiden met andere talen
- Hij is erg uitgebreid
- Goede documentatie over hoe je hem moet gebruiken en integreren

We wilden toen we net begonnen, de editor voornamelijk zelf schrijven. Maar het schrijven van de editor in Javascript, HTML en PHP is toch meer werk geworden dan wij in eerste instantie verwacht hadden. Voornamelijk in browser compatibiliteit gaat veel tijd zitten. De volgende onderdelen vragen grote tijdinvesteringen om opgelost te worden:

1. Javascript functies voor editor benodigdheden.
   (a) Positie bepalen van de cursor.
   (b) Met selecties werken.
2. Layout: niet elke browser rendered de layout het zelfde aan de hand van de zelfde HTML en CSS.

Vanwege tijdgebrek zijn we dus afgestapt van het idee om zelf de editor te schrijven en hebben we besloten om CodeMirror te gebruiken als de basis voor onze editor.

2.4 Documentatie

In het documentatie gedeelte van ons dOPE systeem moet je zowel tekst kunnen schrijven als plaatjes invoegen. Het moet mogelijk zijn een stuk tekst te schrijven over je programma, maar ook moet het mogelijk zijn om een PDF bestand toe te voegen of een plaatje (JPG) kunnen uploaden. We willen ook erg graag dat het mogelijk wordt om UML toe te voegen of zelfs UML in zijn geheel ondersteunen. Door tijdgebrek hebben we besloten om eerst het tekstgedeelte af te krijgen.
We hebben onderzocht welke Documentatie bij ons dOPE systeem past. Ook hebben we rekening gehouden met de programmeertaal, aangezien we de documentatie pas gingen onderzoeken nadat we de programmeertaal hadden bepaald (in tegenstelling tot de editor). Alle documentatie systemen zijn dus in Javascript. De volgende documentatie systemen zouden gebruikt kunnen worden:

1. Etherpad
   Etherpad was open source, maar is overgenomen door Google. De sorce

code is wel beschikbaar, dus misschien kunnen we die hergebruiken.
EtherPad is a web-based collaborative real-time editor, allowing up to sixteen people to edit a text document at the same time, and see all of the participants' edits in real-time, each in their own color. Participants can permanently save revisions at any time, and it provides a separate chat box in the sidebar. Automated markup of JavaScript code was made available shortly after the launch. EtherPad itself is implemented in JavaScript, on top of AppJet, with the real-time functionality achieved through Comet streaming. At the time of its launch, EtherPad was the first web application to achieve true real-time performance, a feat previously only achieved by desktop applications such as SubEthaEdit (for Mac), Gobby or MoonEdit (both multi-platform). Existing web editors at the time could only achieve near-real-time performance.

2. Synchro Edit
   Het programma heeft geen updates meer gehad sinds 2007, maar de source code is wel beschikbaar.
   SynchroEdit is a browser-based simultaneous multiuser editor, a form of same-time, different-place groupware. It allows multiple users to edit a single web-based document at the same time, and it continuously synchronizes all changes so that users always have the same version. SynchroEdit's main editor is fully WYSIWYG, dynamically displaying bolds, italics, underlines, strikethroughs, with various justifications, indents and listing styles as an author inputs them. SynchroEdit also supports a simple, text-only editor for more basic documents. To clarify the multiuser experience, the editor window clearly depicts every user's changes in a specific color and also marks where each user is currently editing with a colored flag listing the user's name.

3. Media Wiki
   MediaWiki is a web-based wiki software application used by all projects of the Wikimedia Foundation, and many other wikis. Originally developed to serve the needs of the free content Wikipedia encyclopedia, today it has also been deployed by companies for internal knowledge management, and as a content management system. Notably, Novell uses it to operate several of its high-traffic websites. MediaWiki is written in the PHP programming language, and can use either the MySQL or PostgreSQLrelational database management system.

4. DokuWiki
   DokuWiki is a standards compliant, simple to use Wiki, mainly aimed at creating documentation of any kind. It is targeted at developer teams, workgroups and small companies. It has a simple but powerful syntax which makes sure the datafiles remain readable outside the Wiki and eases the creation of structured texts. All data is stored in plain text files no database is required.

Onze voorkeur ging uit naar DokuWiki, omdat deze gericht is op ontwikkelaars en op documentaties voor kleine tot middelgrote projecten. Maar toen we de code gingen bekijken, zagen we dat het een standalone systeem is. We hebben al een heel framework, dus veel van de dingen die in DokuWiki zitten krijgen we dan dubbel, aangezien ze ook in ons framework zitten. We konden zelf een interface schrijven die het DokuWiki systeem kan gebruiken of een ander documentatie systeem gebruiken of we moesten zelf een documentatie systeem gaan bedenken.
Nadat we de mogelijkheden om een documentatie systeem te integreren in ons dOPE systeem hebben bekeken, zijn we tot de conclusie gekomen dat het meer werk was om een bestaand systeem gestreamlined te integreren. Dus hadden we besloten om zelf een simpele wiki te schrijven. De vereisten voor de wiki zijn:

1. Pagina's aanmaken
2. Simpele editor met default wiki syntax
3. Enkele andere simpele wiki features

3 Resultaten

Wat wilden we bereiken:

1. Een gebruiksvriendelijke editor, waar in je met andere mensen tegelijk samen kan werken aan een programmeerproject
2. Een compiler om de code uit te voeren
3. Een werkende debugger
4. Een projecten systeem, waar je een overzicht hebt van de projecten waar je mee bezig bent
5. Een documentatie systeem, waar je UML, tekst of plaatjes in kan verwerken
6. Een rechtensysteem, waarbij je een project public of closed source kan maken

Wat hebben we bereikt:

1. Een editor
2. Een projecten systeem waar je een overzicht hebt van de projecten waar je mee bezig bent
3. Een documentatie systeem, waar je tekst in kan verwerken
4. Een rechtensysteem, waarbij je een project public of closed source kan maken

We zijn erg ambitieus geweest toen we besloten om een online programming editor te schrijven. We wilden erg veel zelf schrijven, omdat we op die manier alles in de hand hadden en we onze eigen ideen zo goed mogelijk konden implementeren. Helaas was hier niet genoeg tijd voor en waren we genoodzaakt om delen van andere programma's en systemen te gebruiken.
Het framework was hier het eerste voorbeeld van, we maken nu gebruik van een bestaand framework, genaamd Kohanna. Dit framework zorgt voor een goede basis en is makkelijk uit te breiden.
Hierna was ook de editor aan de beurt en we maken nu gebruik van CodeMirror als de basis voor onze editor. Er zijn nog een heleboel gebruikersvriendelijke functies die we moeten toevoegen, zoals Color Highlighting en Autocompletion, maar vanwege tijdsgebrek hebben we dit achterwege gelaten.
De debugger hebben we wel toegevoegd, maar werkt nog niet, aangezien de compiler er niet bij zit. Er is dus wel al plaats gereserveerd voor de debugger. De compiler hebben we achterwege moeten laten, weer vanwege een tekort aan tijd.
Het documentatie systeem bleek eenvoudiger om zelf te maken, dan een bestaand systeem aan te passen of in zijn geheel te gebruiken.
Ook het rechtensysteem is gelukt om te maken. Een gebruiker kan aangeven of hij een public of closed source project wil maken, op het moment dat hij het project aanmaakt. Een functie om te switchen als je eenmaal een project hebt aangemaakt, is ook toegevoegd.

4 Discussie

Is dOPE beter dan de rest?
Nee, op dit moment niet. Daarvoor hebben we simpelweg te weinig tijd gehad. Maar dOPE combineert wel heel veel elementen die belangrijk zijn voor ons, als programmeurs, in een Online Programming Editor. Deze combinatie is uniek en nog niet te vinden in een andere open source Online Programming Editor. Dit kan nog worden uitgebreid als er meer tijd in wordt gestoken.
Wat is dOPE dan wel?
dOPE is op dit moment de basis van een online programming editor en heeft de mogelijkheid om uit te groeien tot de beste online programming editor op de markt. Vanwege de modulariteit en dus de mogelijkheid om je eigen plugins te schrijven, kan dOPE uitgroeien tot de beste online programming editor.
Waarom zou ik dOPE gebruiken?
dOPE is voornamelijk geschikt voor kleine groepsopdrachten of projecten, waarbij je met meerdere mensen moet programmeren. Sommige mensen kiezen voor SVN als ze aan zo'n opdracht moeten werken. Met SVN kun je bestanden die verandert zijn, bijvoorbeeld de code die je net hebt geprogrammeerd, uploaden naar een server en dan kan iemand anders de nieuwe code downloaden. Met ons dOPE systeem gaat opslaan automatisch en hoef je niet meer een hele map of het hele bestand up te loaden. dOPE is ook geschikt om documentatie in uit te werken, zodat je meer uitleg kan geven over bepaalde functies of keuzes die

gemaakt zijn in je project. Ook is er gedacht aan gebruiksvriendelijkheid, er is gebruik gemaakt van tabbladen en shortkeys voor het inspringen van de code.

5 Conclusie

Met dOPE hebben we een begin gemaakt van een online programming editor. We hebben bestaande systemen en programma's gebruikt en gecombineerd samen met onze eigen ideen. Omdat dOPE modulair is gemaakt, is het project nooit af en zijn uitbreidingen mogelijk. Iedereen die een verbetering toe wil voegen, kan dit indien hij of zij genoeg kennis heeft van Javascript en PHP. Ook als een bepaalde taal nog niet ondersteund wordt, kan deze worden toegevoegd. We hebben op dit moment een stevige basis gelegd waar op voortgebouwd kan worden.

6 Bijlagen

We hebben de documentatie van dOPE online gezet. Deze is **(hier)** te vinden.

References

[1] MichałAntkiewicz, Thiago Tonelli Bartolomei, and Krzysztof Czarnecki. Fast extraction of high-quality framework-specific models from application code. *Automated Software Engg.*, 16(1):101–144, 2009.

[2] L. Peter Deutsch and Butler W. Lampson. An online editor. *Commun. ACM*, 10(12):793–799, 1967.

[3] David W. Embley and George Nagy. Behavioral aspects of text editors. *ACM Comput. Surv.*, 13(1):33–70, 1981.

[4] Glenn E. Krasner and Stephen T. Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system, 1988.