Haaga-Helia
ammattikorkeakoulu Oy

Tehokas Drupal-kehitys: Tekniikat ja työtavat

Minna Lundell

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2015

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

2.5.2015

Tekijä(t)
Minna Lundell

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön otsikko
Tehokas Drupal-kehitys: Tekniikat ja työtavat

Sivu- ja liitesivumäärä
26 + 2

Opinnäytetyön otsikko englanniksi
Efficient Drupal development: Technologies and methods

Opinnäytetyössä tutkittiin erään Drupal-sivustoja toteuttavan yrityksen kehittäjien työssään käyttämiä tekniikoita ja työtapoja.

Tutkimus toteutettiin tapaustutkimuksena "Uudelleenkäytettävien komponenttien tuottaminen ja konventiot yrityksen Drupal-kehityksessä". Tutkimuksen pohjana toimi yrityksen kehittäjille tehty laadullisen tutkimuksen kysely. Tutkimuksen tarkoituksena oli kartoittaa yrityksen Drupal-kehitystyön tämänhetkistä tilaa sekä etsiä mahdollisia kehityskohteita entistä tehokkaamman työnteon takaamiseksi.

Opinnäytetyön teoriataustassa esitellään Drupal-sisällönhallintajärjestelmän modulaarinen rakenne sekä Drupal-kehityksen perusteita. Teoriataustan pääpaino on Drupalin komponenteissa (moduuleissa, teemoissa ja featureseissa), joiden kehitystyöhön liittyviä osa-alueita tutkimuksessa tutkittiin.

Tutkimuksen tuloksista voitiin todeta, että uudelleenkäytettävien komponenttien kehitystyötä tukevat niin konventiot, ajatusmallit kuin dokumentaatiokin, kunhan nämä kaikki ovat jokaisen kehittäjän tiedossa. Kehitysehdotuksista tärkeimpiä olivatkin konventioiden dokumentaation ja kehittäjien välisen viestinnän parantaminen.

Asiasanat
Drupal, komponentti, konventio, uudelleenkäytettävyys

HAAGA-HELIA
University of Applied Sciences

Abstract

2.5.2015

Author(s)
Minna Lundell

Degree programme
Business Information Technology

Report/thesis title	**Number of pages and appendix pages**
Efficient Drupal development: Technologies and methods	**26+2**

This thesis focuses on the technologies and methods the developers of the case company use to create websites with the Drupal content management system.

The study was carried out in the form of a case study named "Development of reusable components and the usage of conventions in a company's Drupal development". The study was based on feedback to a questionnaire targeted the Drupal developers of the company. The purpose of the study was to survey in what state the Drupal development of the company was and what could be improved to increase its efficiency.

The theory chapters of the thesis introduce the modular structure of the Drupal content management system as well as some of the basis of Drupal development. The focus of these chapters, as well as the whole thesis, is on development of components (modules, themes and features) that are used to build Drupal.

The conclusions of the study indicate that development of reusable components can be improved by implementing conventions, paradigms and documentation, as long as every developer has knowledge of them. The most important suggestions for areas to improve include better documentation of the conventions as well as strengthening the communication among the developers.

Keywords
Drupal, component, convention, reusability

Sisällys

1 Johdanto

Sisällönhallintajärjestelmiä on olemassa useita erilaisia, ja eräs suosituimmista ja monipuolisimmista avoimen lähdekoodin sisällönhallintajärjestelmistä on Drupal. Drupalin vahvuudet ovat sen joustavuudessa: Drupal on rakenteeltaan modulaarinen, ja laajennettavissa erilaisilla komponenteilla, kuten teemoilla ja moduuleilla, vastaamaan hyvin monipuolisia teknisiä ja ulkonäöllisiä vaatimuksia.

Vaikka Drupalin toiminnallisuuden laajentamiseen on olemassa valmiina tuhansia komponentteja, saattavat myös Drupalilla verkkosivustoja tekevän yrityksen kehittäjät tuottaa useita omia komponenttejaan jokaista toteutettua verkkosivustoa kohden. Tällöin kehittäjien voisi olla järkevää pohtia tapoja, joilla näitä itse toteutettuja komponentteja voidaan hyödyntää uudelleen parhaalla mahdollisella tavalla. Tähän voidaan pyrkiä sopimalla yhteisiä ajatusmalleja ja konventioita.

1.1 Tutkimusongelma ja tutkimuksen tavoite

Opinnäytetyössä tutkitaan erään Drupal-sivustoja toteuttavan yrityksen kehittäjien työssään käyttämiä tekniikoita ja työtapoja. Sisällönhallintajärjestelmiksi tarkoitetut sivustot seuraavat usein keskenään samankaltaisia rakenteita ja sisältävät samoja toiminnallisuuksia. Tällaisten teemojen tunnistaminen ja hyödyntäminen saattaa auttaa tehostamaan yrityksessä tapahtuvaa kehitystyötä.

Opinnäytetyö perustuu kolmeen tutkimuskysymykseen:

- Minkälaisia komponentteja tuotetaan usein?
- Onko kehitystyössä käytössä konventioita?
- Ovatko komponentit uudelleenkäytettäviä?

Näihin kysymyksiin etsitään vastauksia yrityksen Drupal-kehittäjille suoritetun kyselyn avulla. Tapaustutkimuksen tavoitteena on selvittää, minkälaisia yrityksen kehittäjien työtavat ovat tällä hetkellä, sekä millä tavoilla työn kulkua ja tehokkuutta voidaan nykytilanteesta kehittää.

1.2 Rajaukset

Opinnäytetyön tarkoituksena on tutkia tekniikoita ja työtapoja sekä mahdollisia konventioita, joilla Drupal-verkkosivuston komponentteja tuotetaan. Opinnäytetyön lopputuloksena ei kuitenkaan synny Drupal-verkkosivustoa tai valmiita komponentteja.

Opinnäytetyössä esitelty Drupalin versio on Drupal 7. Opinnäytetyössä esitellyt ohjelmistot ja ohjelmointikielet ovat versioiltaan yhteensopivia Drupal 7:n kanssa. Tarkemmat tiedot yhteensopivista ohjelmistoista ja ohjelmointikielistä löytyvät esimerkiksi sivustolta Drupal.org.

1.3 Käsitteet

API (Application programming interface)
Ohjelmointirajapinta, joka määrittelee, kuinka eri ohjelmat voivat toimia vuorovaikutuksessa keskenään.

Bisneslogiikka
Ohjelmiston kerros, joka käsittelee dataa tietokannan ja käyttäjän välillä.

Esityskerros
Ohjelmiston kerros, jossa data muotoillaan käyttäjän näkemään muotoon.

Features
Drupalin yhteisön tuottama moduuli, jolla voidaan tehdä moduuleita sivuston tietokantaan tallennetuista asetuksista. Tässä opinnäytetyössä Featuresilla tuotettuja moduuleita kutsutaan featureseiksi.

Komponentti
Tässä opinnäytetyössä komponentti on kattokäsite Drupalin moduuleille, teemoille ja featureseille.

Konventio
Yleinen, yhdessä sovittu tapa jonkin asian tekemiseen.

Modulaarisuus
Moduuleista koostuva asia, esimerkiksi ohjelmisto, johon voidaan liittää moduuleita uusien toiminnallisuuksien saavuttamiseksi.

Moduuli
Ohjelmistoon liitettävä, itsenäinen osa, joka sisältää esimerkiksi jonkin tietyn toiminnallisuuden.

Proseduraalinen ohjelmointi

Ohjelmointitapa, jossa ohjelma koostuu pääohjelmasta ja aliohjelmista. Aliohjelmat suorittavat ohjelmassa omaa rajattua toiminnallisuuttaan alusta loppuun. Aliohjelmia voidaan käyttää tarvittaessa uudelleen.

Refaktorointi

Olemassa olevan ohjelmiston koodin parannustyöstä käytettävä termi.

Responsiivinen ulkoasu

Verkkosivuston ulkoasu, joka mukautuu sen tarkasteluun käytetyn laitteen näytön koon mukaisesti.

Sisällönhallintajärjestelmä

Verkkosovellus, jolla kerätään, hallinnoidaan ja julkaistaan sisältöä. Sisällöllä voi olla voi olla useita eri esitysmuotoja ja käyttötapauksia.

Uudelleenkäytettävyys ohjelmistoprojekteissa

Pyrkimys siihen, että ohjelmiston osia voidaan käyttää uudelleen muissa ohjelmistoprojekteissa.

2 Drupal

Tässä kappaleessa esitellään Drupalin historiaa, sekä Drupaliin ja sen komponentteihin liittyviä käsitteitä ja ominaisuuksia.

2.1 Drupalin historia ja sisällönhallintajärjestelmän käsite

Drupalin kehitystyö alkoi vuonna 1999, kun opiskelija Dries Buytaert alkoi kehittämään itselleen ja ystävilleen verkossa toimivaa keskustelupalstaa. Pian useat yksityishenkilöt kiinnostuivat projektista, ja se muutettiin avoimen lähdekoodin projektiksi kenen tahansa kehitettäväksi. Drupal 1.0.0 julkaistiin vuonna 2001. (Drupal.org 2015a.)

Avoimen lähdekoodin ohjelmistoille yhteistä on vapaaehtoisten suorittama jatkuva kehitystyö. Tämä mahdollistaa uusien verkkokehityksen tekniikoiden nopean lisäämisen osaksi sovellusta. Avoimen lähdekoodin ohjelmistojen etuna on myös niiden ilmaisuus. (Nixon 2014, 12.)

Nykyisessä muodossaan Drupal on modulaarinen sisällönhallinta-alusta, jonka ylläpidosta ja kehityksestä vastaa tuhansien käyttäjien ja kehittäjien yhteisö. Drupalin käyttöehdot löytyvät GNU General Public License:sta, joka takaa käyttäjille rajoittamattoman oikeuden ladata ja jakaa Drupalin lähdekoodia ja sen lisäosia. (Drupal.org 2015a.)

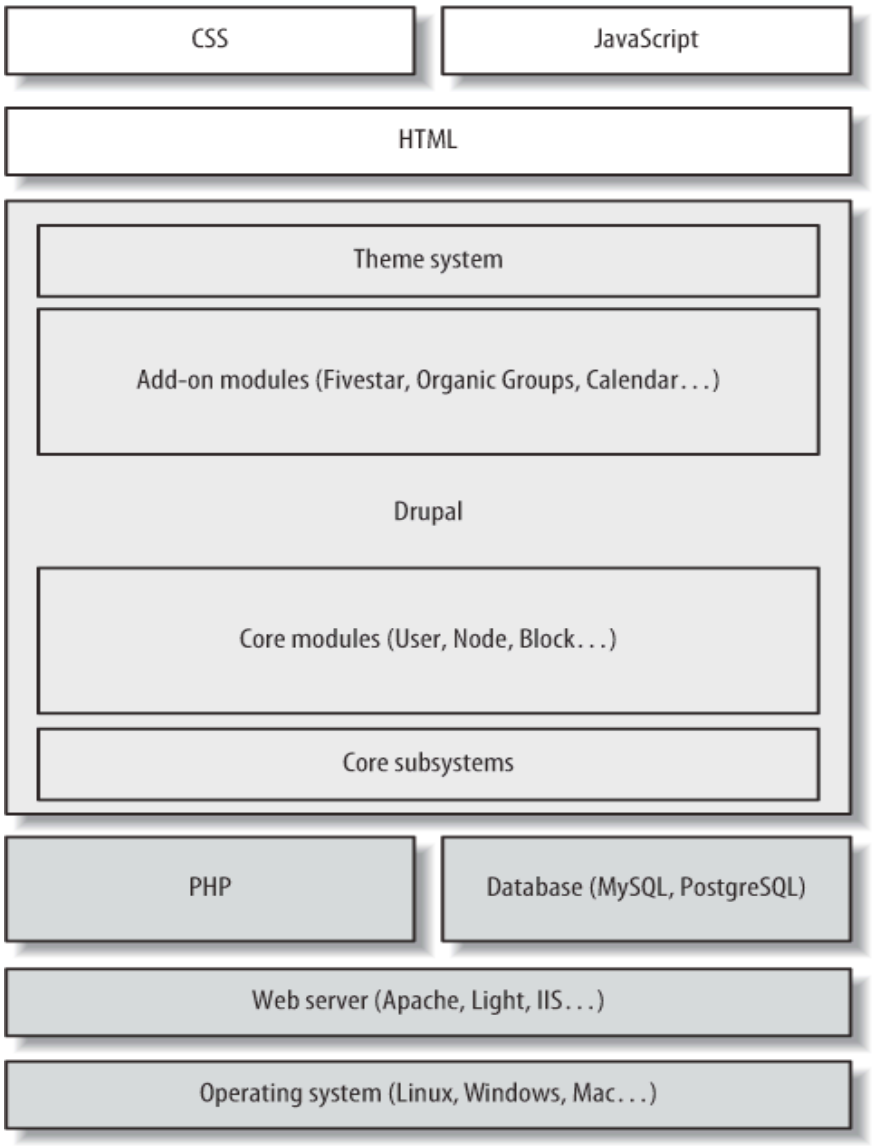
Vaikka Drupalilla on mahdollista toteuttaa hyvin monimuotoisia verkkosivustoja, sen pääasiallinen käyttötapa on sisällönhallintajärjestelmä. Sisällönhallintajärjestelmällä kerätään, hallinnoidaan ja julkaistaan erilaista sisältöä. Parhaimmillaan sisällönhallintajärjestelmällä voidaan esittää kerran kerättyä sisältöä useassa eri muodossa verkkosivuston sisällä ja ulkopuolella esimerkiksi sisältösivuilla tai uutiskirjeissä. (Boiko 2005, 72.)

2.2 Drupalin rakenne

Drupal-verkkosivusto rakentuu Drupalin ytimestä (Drupal core), sekä valinnaisista komponenteista kuten moduuleista (modules) ja teemoista (themes). Drupalin ytimen mukana toimitetaan myös valmiiksi joitakin moduuleita ja teemoja. Drupalin hyödyntämiä teknologioita ovat esimerkiksi PHP, HTML, JavaScript ja CSS. Toimiakseen Drupal-verkkosivuston perusasennus tarvitsee palvelinympäristön, tietokantayhteyden, sekä verkkoselaimen sivuston rakentamiseksi. (Hodgdon 2012, 1-3.)

Nämä eri teknologiat ja niiden päällekkäin ja limittäin asettuvat eri toimintatasot yhdessä tuottavat Drupalin rakenteen, niin sanotun ”Drupal-pinon” (Drupal stack, kuva 1). Pinon

alimmalta tasolta löytyvät palvelinpuolen teknologiat kuten tietokanta ja PHP-tulkki, jotka mahdollistavat Drupal-sivustojen dynaamisen sisällön tuottamisen. Pinon ylimmillä tasoilla ovat asiakaspuolen teknologiat, joita ovat esimerkiksi HTML, CSS ja JavaScript. Ne kuuluvat Drupalin teematasoon eli esityskerrokseen. Drupalin bisneslogiikka sijoittuu pinon keskivaiheille käyttäen PHP:tä ja hyödyntäen tietokantayhteyttä. (Robbins ym. 2012, 8-9.)



Kuva 1. Drupal stack: Drupalin vaatimien ja hyödyntämien teknologien esittely (Robbins ym. 2012, 9.)

Drupal.org-sivusto huolehtii Drupalin ytimien ja komponenttien jakamisesta. Tältä sivustolta löytyvät esimerkiksi Drupalin ytimen suositellut julkaisusarjat. Drupal.org tukee kerrallaan aina kahta viimeisintä julkaisusarjaa, jotka ovat huhtikuussa 2015 Drupal 6 ja Drupal 7. Julkaisusarjan sisällä jokainen versio (7.34, 7.35 jne.) tuo ytimeen joko uusia ominaisuuksia tai turvapäivityksiä. (Drupal.org 2015b.)

Drupal.orgin suositus ytimeksi on tällä hetkellä Drupal 7. Drupal 8:n kehitys on käynnissä, mutta ydin ei ole vielä valmis käytettäväksi tuotannossa sen kehityksen ollessa liian varhaisessa vaiheessa. (Drupal.org 2015b.)

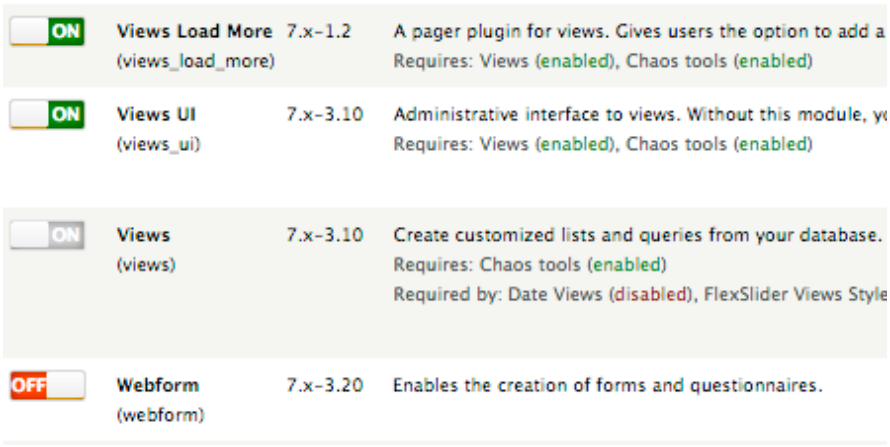
Asentamalla Drupalin ytimen ja sen vaatimat ohjelmistot palvelimelle sivusto on valmis käytettäväksi sisällönhallintajärjestelmänä. Useimmiten sivustolle vaaditaan lisäksi kuitenkin toiminnallisuuksia, joihin pelkkä Drupalin ydin ei kykene. Silloin ytimeen voidaan liittää erilaisia komponentteja, kuten moduuleita ja teemoja. (Hodgdon 2012, 1-3.)

2.3 Moduulit

Moduulit ovat PHP:llä ohjelmoituja komponentteja, jotka laajentavat Drupalin toiminnallisuutta Drupalin API:n kautta. Drupalissa moduulit voidaan jakaa kolmeen kategoriaan (Drupal.org 2015c.):

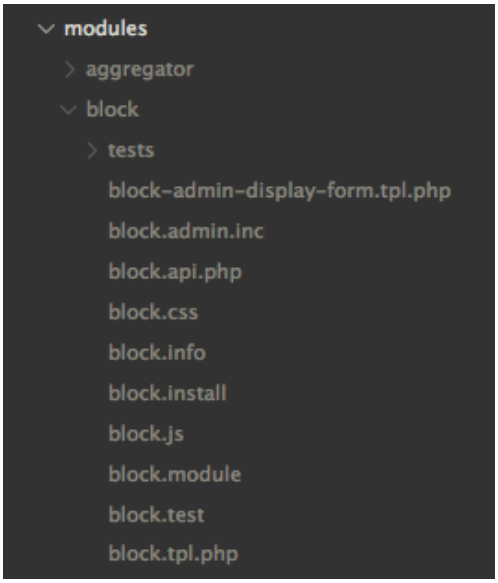
- Drupalin ytimen moduulit (core modules).
- Yhteisön kehittäjien tuottamat moduulit (myöhemmin "moduulit") jotka lisäävät Drupaliin toiminnallisuutta (contributed modules).
- Räätälöidyt moduulit, jotka ovat yksityisten kehittäjien tuottamia vastaamaan heidän omaa tarvettaan (custom modules).

Drupal.org-sivuston moduulilistalta löytyy tuhansia moduuleita Drupalin eri versioille. Moduuleita voi ottaa käyttöön tai niiden asetuksia voidaan muuttaa asennetulta Drupal-sivustolta löytyvästä hallintapaneelista (kuva 2). (Hodgdon 2012, 1-3.)



Kuva 2. Näkymä erään asennetun Drupal-sivuston moduuleista hallintapaneelissa

Drupal-moduuli koostuu yksinkertaisimmillaan kansiosta, joka sisältää .info-tiedoston ja yhden tai useamman PHP:llä kirjoitetun toiminnallisuutta sisältävän tiedoston (kuva 3). .info-tiedosto kertoo Drupalille sen käyttämän Drupalin ytimen sekä PHP:n version, sekä tiedon siitä, mistä tiedostoista moduulin toiminnallisuus löytyy. Selkein toiminnallisuutta sisältävä tiedosto on päätteeltään .module. (Butcher ym. 2010, 25-34.)



Kuva 3. Drupalin ytimeen kuuluvan Block-moduulin kansio ja tiedostorakenne

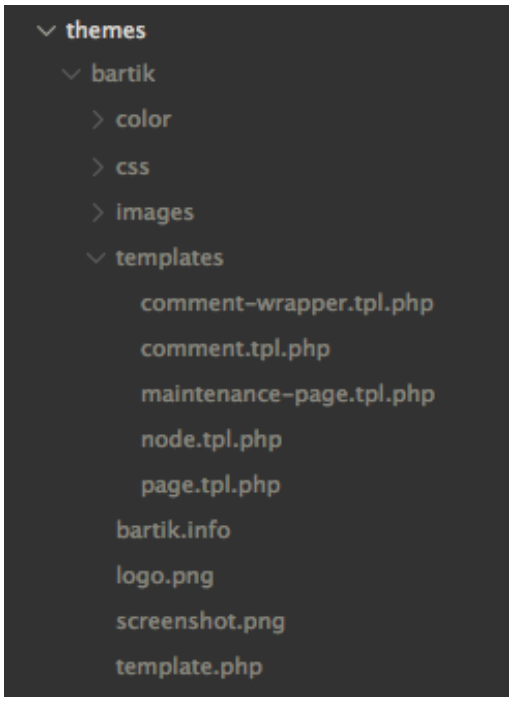
Esimerkki moduulista ja sen toiminnasta Drupalissa on ytimeen kuuluva moduuli nimeltä Block. Blockilla voidaan tehdä lohko, jolle on mahdollista kirjoittaa sisältöä ja määritellä, missä alueessa (region) se sivustolla näytetään. Lohkoja Drupalissa ovat oletuksena esimerkiksi valikot ja hakukentät, jotka voitaisiin määritellä näkymään sivuston otsikkoalueella. Yksinään Block-moduuli on kuitenkin usein liian rajallinen monien verkkosivustojen toiminnallisuuksien toteuttamiseen. Blockin esittelemää lohkojen luomista voidaankin hyödyntää joillakin toisilla moduuleilla. Tällaiset moduulit voivat olla niin yhteisön tai yksityisen kehittäjänkin luomia. Esimerkki Blockia hyödyntävästä moduulista on Views, jolla sivuston sisällöstä voidaan luoda erilaisia esitystapoja näkymälohkoihin (Views block). (Drupal.org 2015d.)

2.4 Teemat

Teemat ovat Drupalin ytimeen lisättäviä sivuston esityskerrokseen vaikuttavia komponentteja. Valmiita teemoja on olemassa paljon, ja niitä voidaan ladata moduulien tapaan Drupal.orgista. Sivuston sisältö, joka on tallennettu tietokantaan, ei ole riippuvainen sivustolla käytössä olevasta teemasta. Teeman vaihtaminen Drupalissa on siis mahdollista ilman sisältöön tehtäviä muutoksia. (Hodgdon 2012, 3.)

Kuten moduulit, myös Drupalin teemat koostuvat .info-, .tpl.php-, CSS-tiedostoista (Sass-, SCSS- ja Less-tiedostot ovat myös mahdollisia) sekä JavaScript-tiedostoista (kuva 4). .info-tiedosto sisältää teeman rakenteen, vaatimukset ja joskus teeman asetuksia. Esimerkiksi teemaan kuuluvat alueet (regions), joihin muun muassa lohkoja voidaan asemoida, esitellään .info-tiedostossa. .tpl.php-tiedostot sen sijaan ovat PHP-pohjaisia mallinnus-

tiedostoja (template files), jotka sisältävät PHP:tä ja HTML:ää. Mallinnustiedostoilla voidaan määritellä niin koko sivuston kuin vain yhden lohkonkin rakenne. (Kumar 2012, 35-38.)



Kuva 4. Drupalin ytimeen kuuluvan Bartik-teeman kansio ja tiedostorakenne

Useimmiten sivuston ulkoasua rakennettaessa pohjana käytetään valmista perusteemaa (base theme), kuten Bartikia, jota muokataan siihen perustuvan aliteeman (subtheme) kautta. Perusteema sisältää siis pohjan, jonka toiminnallisuutta voidaan jatkaa tai muokata aliteemassa. Tämän työtavan etuna on perusteeman toteuttama valmis rakenne, jonka muokkaaminen aliteemassa ei edellytä kokonaan uuden teeman rakentamista. (Crittenden 2014, luku 3.)

2.5 Features

Features on Drupal-yhteisön kehittäjien tuottama moduuli, joka mahdollistaa useimpien Drupal-sivuston tietokantaan tallennettujen asetusten viemisen PHP-koodiksi ja näin versionhallinnan piiriin. Features tuottaa asetuksista moduulin, featuresin, jota voi käyttää kuten muitakin sivustolle asennettuja moduuleita. (Hodgdon 2012, 38.)

Featuresin tuottamia featureseja voidaan käyttää eri Drupal-asennuksissa vähentäen samojen asetusten tekemistä useita kertoja. Featuresin ongelma on kuitenkin sen yhteensopimattomuus joidenkin muiden yhteisön tuottamien moduulien asetusten kanssa, jolloin niiden vienti featuresiksi ei onnistu. (Hodgdon 2012, 38.)

Featureseiksi on kannattavaa tehdä rajattuja, useita pieniä kokonaisuuksia suurten sijaan. Näin eri featuresien välille syntyy mahdollisimman vähän riippuvuuksia ja niiden uudelleenkäyttö pysyy mahdollisena. Esimerkki featuresista voisi olla vaikkapa yhden sisältötyypin (content type) asetukset. (Falk 2011, 294-298.)

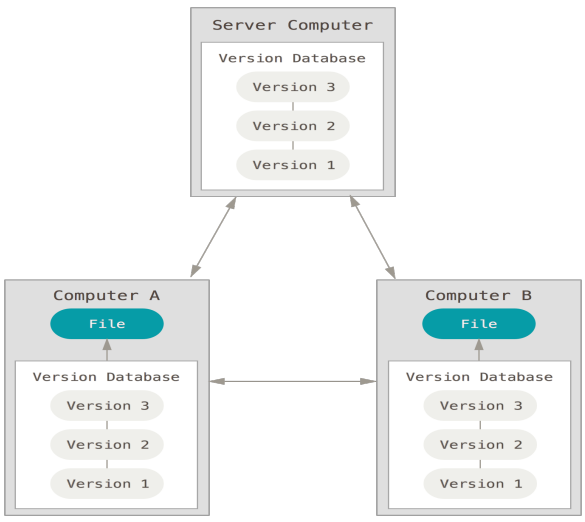
3 Drupal-kehityksen perusteita

Drupal hyödyntää useita eri teknologioita tuottaakseen täysin muokattavissa olevan sisällönhallintajärjestelmän. Tässä kappaleessa esitellään versionhallinnan käsite ja Git, sekä aiemmin mainitut teknologiat kuten PHP, HTML, CSS ja CSS-esikäsittelijät sekä JavaScript niiltä osin, kuin ne ovat moduulien ja teemojen kehittämisen ja käyttämisen kannalta olennaisia.

3.1 Versionhallinta

Versionhallinnan työkaluja ovat versionhallintajärjestelmät (version control systems). Versionhallinnalla pidetään kirjaa tiedostoihin tehdyistä muutoksista versioimalla ne. Jos esimerkiksi johonkin tiedostoon tehdyt uudet muutokset eivät kelpaa enää osaksi ohjelmistoa, ne voidaan hylätä palaamalla tiedoston aiempaan versioon. Versionhallinnan avulla voidaan myös seurata sitä, milloin jokin virhe on ilmennyt ohjelmistossa ensimmäisen kerran. Versionhallintajärjestelmien tehtävänä on myös mahdollistaa nopea palautuminen tilanteesta, jossa vaikkapa kehittäjän huolimaton työ tuhoaa jonkin tiedoston tämän kehitysympäristössä. (Chacon & Straub 2014, 13-18.)

Eräs ohjelmistokehityksessä suosittu versionhallintajärjestelmä on Git. Git on jaetun versionhallinnan järjestelmä (Distributed Version Control System, DVCS), jossa jokainen kehittäjä voi työskennellä toisistaan riippumatta projektin yhden saman tietolähteen (repository) parissa. Tietolähdettä voidaan säilyttää palvelimella esimerkiksi sivustoilla kuten GitHub ja BitBucket, jolloin se on helposti kaikkien kehittäjien saatavilla (Kuva 5). (Chacon & Straub 2014, 13-18.)



Kuva 5. Jaetun versionhallinnan järjestelmän toiminnan kuvaus (Chacon & Staub 2014, 18).

Drupal-sivustosta versionhallinnan piiriin on suositeltua ottaa lähes kaikki tiedostot sivuston juurihakemistosta alkaen. Poikkeuksen tekevät sivuston loppukäyttäjien lisäämät tiedostot ja tietokanta, jotka voivat muuttua jatkuvasti käytössä olevalla sivustolla ja vaativat versionhallinnan sijasta palvelimelle jonkinlaisen varmuuskopiointijärjestelmän. (Drupal.org 2015e.)

3.2 Ohjelmointi

Drupalin pääasiallinen ohjelmointikieli on PHP. ”PHP: Hypertext Preprocessor”, on web-ohjelmoinnissa yleisesti käytetty avoimen lähdekoodin palvelinpuolen ohjelmointikieli. PHP:tä voidaan käyttää ohjelmointiin itsessään tai sitä on mahdollista upottaa HTML:n sekaan ja näin tuottaa verkkosivustoille dynaamista sisältöä. (PHP.net 2015.)

PHP:n ohjelmointityyli Drupalissa on pääasiassa proseduraalinen (procedural programming). Proseduraalisen ohjelmointityylin takia useat Drupalin moduuleista koostuvat kokoelmasta peräkkäin suoritettavia funktioita. Toisaalta myös olio-ohjelmointia käytetään joissakin Drupalin osissa tarpeen mukaan. (Butcher ym. 2010, 9.)

Moduulien ohjelmallinen rakenne

Drupalin moduulit koostuvat yhdestä tai useammasta PHP-funktioita sisältävästä tiedostosta. Osa näistä funktioista voi olla moduulin omia, mutta moduulin on myös mahdollista hyödyntää kaikkia Drupalin ytimessä valmiiksi määriteltyjä funktioita käyttämällä ”koukkuja” (hook). Moduulit voivat tämän lisäksi määritellä myös omia koukkujaan muiden moduulien käytettäväksi. (Drupal.org 2015f.)

Drupalin koukut ovat funktioita, jotka seuraavat määriteltyä nimeämiskäytäntöä. Koukuista puhutaan ja kirjoitetaan usein tyylillä koukku_nimi (hook_name), jossa ”koukku” on sijaissana sitä käyttävän moduulin nimelle, ja ”nimi” itse koukun nimi. Kun moduuli on esitellyt koukun, Drupalin ydin suorittaa siihen liittyvän toiminnallisuuden, kun jokin tätä koukkua hyödyntävä tapahtuma (event) käynnistyy. Esimerkki tapahtumasta ja sen koukusta voi olla vaikkapa käyttäjän sisäänkirjautuminen (taulukko 1). Lista Drupalin ytimen koukuista löytyy sivustolta Api.drupal.org. (Tomlinson & VanDyk 2010, 33-55.)

Taulukko 1 Käyttäjän sisäänkirjautumiseen liittyvä tapahtuma ja sen koukun nimeämiskäytäntö moduulissa (Tomlinson & VanDyk 2010,33-55)

Tapahtuma	Koukku	Moduuli	Koukun nimi moduulissa käytettäessä
Käyttäjän sisäänkirjautuminen (Login User)	hook_user_login	mytestmodule	mytestmodule_user_login

Ohjelmointistandardit

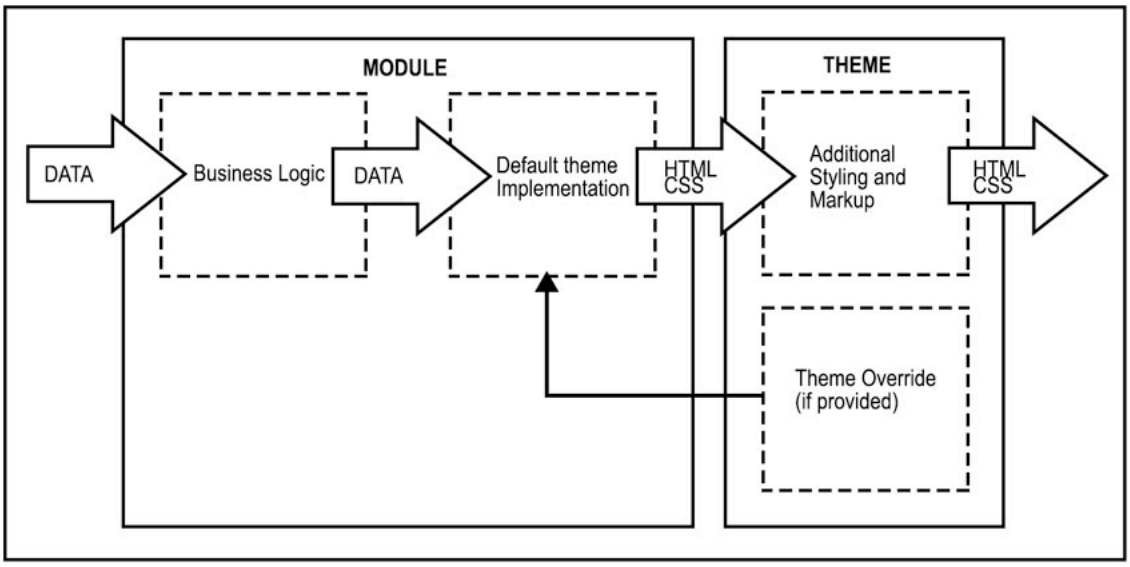
Drupalin ohjelmointistandardit sen ytimen osalta ovat hyvin tiukat, ja myös suurin osa yhteisön tuottamista komponenteista seuraa näitä konventioita. Nämä ohjelmointistandardit päivittyvät jatkuvasti, ja paras dokumentaatio niille löytyykin Drupal.org-sivuston osiosta ”Coding Standards” (ohjelmointistandardit). Ohessa eräitä esimerkkejä Drupalin PHP:n ohjelmointistandardeista (Butcher ym. 2010, 35-37.):

- Sisennyksien on oltava aina kahden välilyönnin (spaces) mittaisia.
- Tyhjämerkkejä (whitespace) käytetään runsaasti luettavuuden parantamiseksi.
- PHP-tiedostoissa ei käytetä sulkevaa tunnistetta.
- Kommentoinnille on omat konventionsa, ja kommenttien on seurattava englannin kielioppia.

Drupalin ohjelmointistandardit pyrkivät parantamaan koodin luettavuutta ja ymmärrettävyyttä, mikä on erityisen tärkeää avoimen lähdekoodin projekteissa, joissa kehitystyö tapahtuu usean kehittäjän toimesta. Avuksi standardien seuraamisessa voi käyttää moduulia Coder, joka ilmoittaa mahdollisesta standardienvastaisesta koodista. (Drupal.org 2015g.)

3.3 Teemataso

Drupalissa esityskerroksen (presentation layer) tuottaminen tapahtuu dynaamisesti teematasossa (theme layer). Teemat ovat Drupal-kehittäjälle näkyvin osa Drupalin teematasoa. Jotta teemataso voi toimia tarkoitetulla tavalla, moduulien tulosteelle suorittaman bisneslogiikan (business logic) ja esitystavan (presentation) on oltava toisistaan erillään. Drupalin teematason ideana on mahdollistaa kaiken tulosteen täysi muokattavuus ja ylikirjoitettavuus. (kuva 7). (Butcher ym. 2010, 61-65.)



Kuva 6. Tulosteen (data) teemoituksen eri tasot: Moduuli määrittelee tulosteen oletusteemoituksen (default theme implementation), jota teema voi joko muokata tai ylikirjoittaa (Butcher ym. 2010, 63.)

Moduulissa tuloste voidaan rekisteröidä teemoitettavaksi esittelemällä se teemakoukuksi (theme hook) theme()-funktion avulla. Theme()-funktiolle voidaan antaa parametreiksi haluttavan teemoitustyylin nimi (esimerkiksi item_list) sekä teemoitettavan tulosteen sisältävä muuttuja. Teemakoukku voi olla joko sivupohjatiedosto (template file) tai teemafunktio. Teemakoukku voidaan toteuttaa vain kerran, joten sivupohjatiedosto ja teemafunktio eivät voi olla käytössä samanaikaisesti. (Butcher 2010, 46-67.)

Sivupohjatiedostot, teemafunktiot ja esikäsittelyfunktiot

Sivupohjatiedostot koostuvat HTML-rakenteesta, johon on upotettu PHP-lausekkeita (PHP statements). Sivupohjatiedosto on esimerkiksi page.tpl.php, jossa määritellään, mitä Drupal-sivun HTML:n body-elementin sisällä tulostetaan. Teemafunktio sen sijaan on PHP-funktio, jossa tuloste muotoillaan. Sivupohjatiedostojen ja teemafunktion suurin ero on niiden hahmotettavuudessa: ensimmäinen muistuttaa ulkonäöltään HTML-sivua, ja toinen yksittäistä funktiota. Drupalissa oletus teemakoukuksi on teemafunktio, mutta sivupohjatiedoston löytyessä järjestelmä käyttää sitä teemafunktion sijasta. Jos sivupohjatiedostojen ja teemafunktioiden saamien muuttujien sisältämä tuloste vaatii muokkausta ennen teemakoukun toteutusta, muokkaus on tehtävä erillisessä esikäsittelyfunktiossa (preprocess function). (Butcher ym. 2010, 61-72.)

CSS ja CSS-esikäsittelijät

Drupalin teematasoon kuuluvat myös moduulien ja teemojen CSS-tiedostot. Kaikki ytimen ja moduulien CSS-tiedostot ovat ylikirjoitettavissa teemassa.
(Drupal.org, 2014.)

CSS-esikäsittelijöitä kuten Sass ja Less voidaan käyttää CSS:n tuottamiseen. CSS-esikäsittelijät mahdollistavat joustavamman CSS:n kirjoittamisen omilla tiedostopäätteillään (.sass/.scss ja .less), jotka käännetään selainten ymmärtämään CSS-päätteeseen. Esikäsittelijöiden ominaisuuksia ovat esimerkiksi muuttujat, mixin-kirjastot kuten Sass:in Compass, sekä mahdollisuus sisäkkäisiin sääntöihin.
(Crittenden 2014, luku 3.)

JavaScript ja jQuery

Suuri osa Drupalin JavaScriptistä hyödyntää jQuery-kirjastoa, joka toimitetaan Drupalin ytimen mukana. JavaScriptiä on mahdollista käyttää Drupalissa esittelemällä .js-tiedostot komponenttien .info-tiedostoissa sivustonlaajuisesti, tai funktiossa drupal_add_js(), jolla skriptin lisäys voidaan tehdä määritellyssä kontekstissa. Drupal käyttää jQueryn konfliktoimatonta tilaa (jQuery.noConflict), joka tulee ottaa huomioon jQueryllä ohjelmoitaessa.
(Butcher 2010, 285-292.)

4 Uudelleenkäytettävien komponenttien tuottaminen ja konventiot yrityksen Drupal-kehityksessä

Drupalilla toteutetut sisällönhallintaan käytettävät verkkosivustot ovat usein rakenteiltaan ja toiminnallisuuksiltaan keskenään suurelta osin samanlaisia. Kehittäjien työtä voidaan helpottaa tunnistamalla samankaltaisuuksia uudelleenkäytettäviksi ja hyödyntämällä yhteisiä konventioita komponenttien kehityksessä.

Opinnäytetyön tapaustutkimus (case) perustuu tietoon siitä, että yrityksen työntekijät käyttävät jo työssään joitakin konventioita joiden tarkoituksena on tukea niin komponenttien kehitystyötä, ylläpitoa kuin uudelleenkäyttöäkin. Tutkimuksessa pyritään hahmottamaan, mikä on yrityksessä tapahtuvan komponenttikehityksen nykytila, sekä löytämään mahdollisia kehityskohteita. Kehitysehdotusten avulla pyritään entistä tehokkaampaan Drupal-kehitykseen.

4.1 Tutkimus- ja analysointitapa

Tutkimuksen toteutustavaksi valikoitui tapaustutkimus uudelleenkäytettävien komponenttien tuottamisesta ja konventioista yrityksen Drupal-kehityksessä. Tiedon keräämiseksi yrityksen työntekijöille tehtiin laadullisen tutkimuksen kysely. Sähköpostilla lähetetty kysely koostui kuudesta avoimesta kysymyksestä, joihin kehittäjiltä odotettiin heidän omiin kokemuksiinsa perustuvia vastauksia. Kysely löytyy opinnäytetyön liitteestä 1.

Kyselyn kysymykset perustuivat seuraaviin tutkimuskysymyksiin:
1. Minkälaisia komponentteja tuotetaan usein?
2. Onko käytössä konventioita?
3. Ovatko komponentit uudelleenkäytettäviä?

Kuhunkin tutkimuskysymykseen liittyi kyselyssä kaksi kysymystä. Tulosten analysointia varten eri tutkimuskysymykseen liittyvät kysymykset lajiteltiin omiksi teemoikseen. Teemoja olivat laatu, uudelleenkäytettävyys ja konventiot (liite 2). Vastausten analysoinnissa hyödynnettiin kaikkien saatujen vastausten lajittelua taulukkomuotoon. Vastaukset tiivistettiin ja merkittiin taulukkoon taustaväreittäin vastaajan mukaan, jotta mahdolliset syy- ja seuraussuhteet eri teemojen välillä olisivat helposti nähtävissä.

Vastausten analysoinnissa päädyttiin käyttämään induktiivista päättelyä. Induktiivisen päättelyn tarkoituksena on johtaa yhdestä tai useammasta yksittäistapauksesta yleistyksen kautta teoria. Induktiivista päättelyä on kritisoitu sen tulosten suppeuden takia, sillä

sen antamat vastaukset ovat täysin riippuvaisia tutkitusta joukosta, eivätkä näin ollen kuitenkaan sovi yleistettäväksi koskemaan kaikkia tapauksia. (Virtuaaliammattikorkeakoulu 2015.)

Tähän tutkimukseen, jossa tutkimustavaksi oltiin valittu tapaustutkimus yhden yrityksen kehitystyöstä ja tutkittavana oli vain pieni perusjoukko (alle kymmenen henkeä), induktiivinen päättely sopi kuitenkin hyvin. Kyselyn vastaukset analysoitiin kolmessa kierroksessa: vastausten raa'asta tiedosta yksinkertaistettuna taulukkoon, taulukosta auki kirjoitetuksi kappaleeseen 4.2 ja lopulta yhteenvedoksi kappaleessa 4.3. Yhteenvedosta tehdyt johtopäätökset, teoria, löytyy kappaleesta 5.1.

Jotta kyselyn avoimiin kysymyksiin saatiin tutkimuksen kannalta olennaisia vastauksia, kysymysten selkeyteen kiinnitettiin erityistä huomiota. Kyselyn yhteydessä vastaajille esiteltiin opinnäytetyössä käytetyt termit "komponentti" sekä "konventio" väärinymmärrysten välttämiseksi.

4.2 Vastausten analysointi

Seuraavassa kappaleessa kyselyn vastaukset on käyty läpi ja kirjoitettu auki. Vastauksia on yleistetty valitun analysointitavan mukaisesti.

4.2.1 Laatu

Kyselyssä tutkittiin kehittäjien tuottamien komponenttien laatua. Kyselyn ensimmäisessä laatuun liittyvässä kysymyksessä "Mitä" pyydettiin tietoa tuotettujen komponenttien laadusta viimeisimmän puolen vuoden ajalta. Komponenttien laaduiksi oli kysymyksessä määritelty featuresit, räätälöidyt moduulit ja teemat (ja aliteemat). Vastauksia pyydettiin viimeisen puolen vuoden ajalta, koska siinä ajassa kehittäjä on todennäköisesti ehtinyt työskennellä monipuolisesti useamman projektin parissa.

"Minkälaisia"-kysymyksessä pyrittiin selvittämään, toistuvatko sisällöiltään tai toiminnallisuuksiltaan samat komponentit useiden kehittäjien työssä. Tällä kysymyksellä odotettiin selviävän, mitä mahdollista samaa työtä kehittäjät tekivät tietämättä aiemmista tai toistensa toteutuksista.

Featuresit

Kaikki kehittäjät vastasivat tuottaneensa featureseja viimeisen puolen vuoden aikana. Jos kehitettävälle sivustolle asennetaan Features-moduuli, se yleensä määrittelee kehityksen

työnkulun perustumaan featuresien käyttöön. Tässä työnkulussa suuri osa sivuston tietokantaan tehdyistä asetuksista paketoidaan featureseiksi versionhallinnan piiriin; featureseja ovat silloin esimerkiksi sisältötyypit, näkymät, kontekstit, asetukset ja käyttö-oikeudet.

Featuresit olivat myös kaikkien paitsi yhden kehittäjän usein tuottamien komponenttien listalla. Selkeästi toteutetuimpia olivat sisältötyyppien ja näkymien featuresit. Koska sivustot ovat sisällönhallintajärjestelmiä, sisällön tallentamiseen (sisältötyypit) ja sisällön eri esitystapoihin (näkymät) liittyvien featuresien yleisyys ei ole yllättävää. Sisältötyyppien ja näkymien yhdessä tuottamia kokonaisuuksia, joita kehittäjät kertoivat toteuttavansa usein, ovat esimerkiksi Uutis- ja Blogi-kokonaisuudet.

Räätälöidyt moduulit

Osa kehittäjistä kertoi tuottavansa myös räätälöityjä moduuleita. Nimensä mukaisesti näillä moduuleilla vastataan johonkin erityiseen tarpeeseen, joka on usein projektikohtainen. Integraatiot asiakkaiden taustajärjestelmiin erilaisten rajapintojen kautta mainittiin esimerkkinä räätälöidyistä moduuleista. Eräällä kehittäjällä räätälöidyt moduulit olivat myös hänen useimmin tuottamiaan.

Teemat ja aliteemat

Räätälöityjä teemoja ei ollut toteuttanut kuin yksi kyselyyn vastanneista. Useimmat kehittäjät olivat kuitenkin toteuttaneet aliteemoja erilaisista valmiista teemoista, kuten Omegasta ja Bootstrapista. Kaikille mainituille teemoille yhteistä on niiden modulaarinen CSS- tai Sass-rakenne sekä responsiivisuus. Myös joitakin räätälöityjä JavaScript- ja jQuery-toiminnallisuuksia oli toteutettu projekteihin viimeisen puolen vuoden aikana. Yhden kehittäjän useimmin toteuttamat komponentit olivat aliteemoja ja niihin liittyvää teemoitustyötä.

4.2.2 Konventiot

Konventioiden tarkoituksena on helpottaa kehittäjien työtä asettamalla kaavoja sille, miten jokin asia tulee toteuttaa. Kyselyssä selvitettiin, mitä konventioita kehittäjät työssään jo käyttävät, ja mitkä uudet konventiot saattaisivat entisestään parantaa työn kulkua.

Versionhallinta

Kehittäjät käyttävät työssään versionhallintaa, ja sen työkaluna Gitiä. Versionhallinta miellettiin kyselyssä yhdeksi konventioksi. Gitin käytössä on kuitenkin mahdollista seurata

erilaisia työnkulkuja ja konventioita (esimerkiksi nimeämiskäytäntöjä), jotka tulee sopia kehittäjien kesken. Mainittuja työnkulkuja olivat ”rebase” (uudelleenalustus) sekä ”merge” (yhdistäminen), jotka ovat hieman toisistaan poikkeavia tapoja alustaa kunkin kehittäjän omassa kehitysympäristössä oleva projekti palvelimen tietolähteen tasalle.

Ohjelmointistandardit ja dokumentaatio

Räätälöityjä moduuleita tuottavat kehittäjät seuraavat vastausten mukaan Drupalin ohjelmointistandardeja. Näihin standardeihin kuuluvat myös useiden samojen kehittäjien mainitsema kommentoimalla tapahtuva dokumentaatio. Moduulien ja niiden yksittäisten funktioiden kommentoinnin avulla kehittäjät eivät välttämättä tarvitse avukseen ulkoista dokumentaatiota. Eräs kehittäjä mainitsee dokumentoinnin avuksi myös ”readme”-tiedostot, joissa voidaan kertoa esimeriksi komponentin ominaisuudet sekä käyttöohjeet.

Nimeämiskäytännöt

Kaikki kehittäjät kertovat käyttävänsä kehitystyössään yhtä tai useampaa nimeämiskäytäntöä. Nimeämiskäytännöt esimerkiksi funktioille on määritelty Drupalin ohjelmointistandardeissa. Kehittäjillä on käytössään myös keskenään sovittuja nimeämiskäytäntöjä esimerkiksi featuresien osalta. Uudeksi konventioksi ehdotettiin omaa nimeämiskäytäntöä kentille. Esimerkiksi sisältötyypit koostuvat usein monista kentistä, joiden valmiiksi sovitut nimeämiskäytännöt vähentäisivät nimien pohtimiseen käytettyä aikaa.

Muita konventioita

Ohjelmiston osien kehittäminen lähtökohdista modulaarisuus ja geneerisyys mainitaan erään kehittäjän seuraamina konventioina. Molemmat konventiot tähtäävät mahdollisimman helppoon komponentin uudelleenkäytettävyyteen. Tämänlaisen kehitystyön ajatusmallina tunnetaan esimerkiksi saman kehittäjän mahdolliseksi uudeksi konventioksi mainitsema DRY (Don't Repeat Yourself). Vaikka vain yksi kehittäjä mainitsi nämä konventioikseen, useiden muiden kehittäjien muiden kysymysten vastauksista selvisi myös heidän pyrkivän samankaltaiseen ajatteluun.

Kehittäjien kehitysehdotukset

Aiemmin mainittujen DRY:n ja kenttien nimeämiskäytäntöjen lisäksi parannusehdotukseksi esitettiin featuresien rajaamisen selkeytystä: kuinka suuri kokonaisuus sopii yhteen fea-

turesiin pakattavaksi. Muuten kehittäjät kokivat mainitsemansa konventiot riittäviksi kehitystyön tukena.

4.2.3 Uudelleenkäytettävyys

Kyselyssä kysyttiin myös, voidaanko komponentteja käyttää uudelleen, ja miksi kyllä tai ei. Tällä kysymyksellä pyrittiin selvittämään mitä hyviä ja huonoja puolia komponenttien kehityksestä löytyy. Viimeisessä kysymyksessä haluttiin selvittää, tukevatko edellisessä kappaleessa käsitellyt käytössä olevat konventiot kehittäjien mielestä komponenttien uudelleenkäytettävyyttä.

Erilaatuisten komponenttien uudelleenkäytettävyys

Uudelleenkäytettävyyden perustana pidettiin geneerisyyttä, joka löytyi jokaisen annetun vastauksen pohjalta. Jos komponentti on rakenteeltaan alunperin riittävän geneerinen, se voidaan ottaa käyttöön useassa projektissa ja sitten räätälöidä vastaamaan projektin tarvetta. Esimerkiksi featuresien kanssa olisi mahdollista toteuttaa useita erilaisia paketteja vaikkapa yleistetyistä asetuksista, ja käyttää näitä uusien sivustojen toteutuksen pohjana.

Sivustojen teemoissa voidaan myös käyttää uudelleen aiemmin toteutettuja aliteemoja tai teemoituksen osia, kun hyödynnetään modulaarista Less-,Sass- tai CSS-rakennetta. Tällöin toteutetaan yksi Less-, Sass- tai CSS-tiedosto sivuston yhdestä osasta, kuten Uutisnäkymän teemoituksesta. Myös tässä tapauksessa tarpeeksi geneerinen toteutus tukee uudelleenkäytettävyyttä.

Räätälöityjen moduulien osalta uudelleenkäytettävyys oli vastausten mukaan vaikeaa toteuttaa. Joskus refaktoroinnilla voisi olla mahdollista hyödyntää räätälöityä moduulia uudelleen.

Uudelleenkäytettävyyden haasteita

Vaikka komponenteissa pyritään uudelleenkäytettävyyteen, se on kehittäjien mukaan välillä haastavaa. Pyrittäessä mahdollisimman suureen geneerisyyteen saatetaan törmätä ongelmaan, jossa jotkin asiat on helpompi toteuttaa aina uudelleen projektikohtaisesti. Tasapainottelu geneerisyyden tuottaman hyödyn ja haitan välillä on siis vaikeaa. Usein yksittäisen ratkaisun tuottaminen nopeasti on kehittäjän ainoa vaihtoehto; esimerkiksi räätälöityjen moduulien kohdalla ajankäyttö on usein rajattu niin, että vain yhden projektissa toimivan toteutuksen tekemiseen kuluva aika on huomioitu aikataulutuksessa.

Featuresien osalta ongelmana on se, että kaikkien yhteisön tuottamien moduulien asetusten paketointi featureksi ei ole mahdollista, jolloin osa työstä on kuitenkin tehtävä käsin, eikä koko sivustoa ole mahdollista toteuttaa vain käyttämällä uudelleen featureseja. Myös sisältötyypit, jotka saattavat riippua esimerkiksi integraatioista, ovat helposti muutamaa poikkeusta lukuun ottamatta aina projektikohtaisia.

Konventiot ja uudelleenkäytettävyys

Yleisesti kaikki kehittäjät pitivät konventioita kehitystyötä tukevana osana uudelleenkäytettäviä komponentteja kehitettäessä. Erään kehittäjän mukaan joitakin räätälöityjen moduulien ja featuresien nimeämiskäytäntöjä pitäisi kuitenkin tarkentaa uudelleenkäytettävyyden parantamiseksi.

4.3 Vastausten yhteenveto

Kun sivustojen kehitys perustuu Features-moduulin käyttöön, jokaiselle sivustolle tuotetaan huomattava määrä enemmän featureseja kuin muita komponentteja. Featureseista yleisimpiä ovat sisältötyyppien sekä sen eri näkymien featuret, jotka ovat yhteydessä toisiinsa.

Räätälöityjen moduulien toteuttaminen on täysin projektikohtaista, eikä aina edes välttämätöntä. Teema-asennus tehdään projektissa vain kerran, mutta responsiivisen sekä modulaarisen CSS-, Sass- tai Less-teeman käyttämisellä voidaan hyödyntää aiemmin tehtyä valmista teemoitusta.

Kaikki kehittäjät toteuttavat työssään featureseja. Muiden komponenttien osalta vastauksissa oli jonkin verran eroja. Näiden erojen perusteella joidenkin kehittäjien voitiin todeta keskittyvän enemmän ohjelmointiin (räätälöityjen moduulien kehitys) tai teemoitukseen (teemakerroksen toteutus).

Kehittäjien yhdessä käyttämiä konventioita ovat versionhallinnan konventiot sekä nimeämiskäytännöt, jotka koskevat kaikkia komponentteja. Dokumentaation ja kommentoinnin mainitsivat featuresien lisäksi räätälöityjä moduuleita toteuttavat kehittäjät. Nämä kehittäjät pyrkivät myös työssään seuraamaan Drupalin ohjelmointistandardeja. Kehittäjillä saattoi olla käytössään myös omia konventioitaan, jotka eivät ole muiden kehittäjien tiedossa.

Useimmat työntekijät olivat tyytyväisiä yrityksessä tällä hetkellä käytettyihin konventioihin. Uutena konventiona ehdotettiin DRY-ajatusmallia.

Uudelleenkäytettävyys perustuu kehittäjien mielestä mahdollisimman suureen komponenttien geneerisyyteen. Räätälöityjen moduulien kohdalla uudelleenkäytettävyys ei aina ole vaadittu ominaisuus. Joskus on myös pohdittava geneerisen toteutuksen vaatimaa aikaa verraten sitä yksittäisen, projektikohtaisen ratkaisun kehittämiseen.

Konventioiden käyttämisen koettiin kehittäjien mukaan tukevan ajatusta uudelleenkäytettävien komponenttien toteuttamisesta, vaikka joitakin konventioita olisi suurimman hyödyn saamiseksi paranneltava.

5 Pohdinta

Tutkimuksessa pyrittiin löytämään vastaus tutkimuskysymyksiin minkälaisia komponentteja kehittäjät tuottavat usein, minkälaisia konventioita he käyttävät, ja pystytäänkö komponentteja käyttämään uudelleen. Tutkimuksen pohjana toimivat kyselyn vastaukset joita analysoitiin kierroksittain kunnes päädyttiin tuloksissa esiteltäviin johtopäätöksiin eli teoriaan.

5.1 Johtopäätökset

Yrityksen kehittäjät käyttävät kehitystyössään useita erilaisia konventioita, jotka tukevat heidän työntekoaan. Konventioilla voidaan määritellä valmiiksi asioita, jotka muuten vaatisivat kehittäjiltä pohdintaa. Kun pohdinnan taakka on konvention avulla poistettu kehittäjältä, tämä voi keskittyä esimerkiksi komponentin nimeämisen sijasta komponentin toiminnallisuuden toteuttamiseen. Konventioiden etuna on myös, että tuntemalla konvention kehittäjä voi ymmärtää myös sen avulla kehitettyjä komponentteja. Näin komponenttien kehitystä on mahdollista tehostaa.

Konventiot kuten ohjelmointistandardit helpottavat komponenttien ylläpitoa, jatkokehitystä sekä uudelleenkäytettävyyttä. Drupalin valmiiksi esittelemiä kattavia ohjelmointistandardeita on järkevää hyödyntää räätälöityjen moduulien kehityksessä.

Dokumentaation tärkeyttä osana kehitystyötä ei pidä unohtaa. Helpoin tapa dokumentoida koodia on kommentoimalla. Kun esimerkiksi jokaisen räätälöidyn moduulin ominaisuudet ja funktioiden kuvaukset löytyvät räätälöidyn moduulin lähdekoodista, dokumentaatio on aina sen parissa työskentelevän kehittäjän saatavilla.

Nimeämiskäytäntöjä kannattaa sopia kaikille komponenteille ja niiden osille, kuten räätälöidyille moduuleille, sisältötyypeille ja kentille. Nimi on esimerkiksi featuresin tapauksessa parhaita tapoja dokumentoida sen ominaisuuksia. Nimeämiskäytäntöjen sopiminen ja niiden käyttäminen parantaa komponenttien hahmottamista.

Komponenttien uudelleenkäytettävyyden parantamiseen voidaan vaikuttaa omaksumalla uusia ajattelutapoja. Komponenttien uudelleenkäyttäminen on mahdollista, jos uudelleenkäytettävyyden tarve huomioidaan jo kehitystyön alkaessa. On kuitenkin tärkeää myös tunnistaa uudelleenkäytettäviksi sopivia komponentteja, sillä kaikkien komponenttien kohdalla uudelleenkäytön huomioiminen ei ole oleellista.

Jos komponentin tuottaminen geneeriseksi vaatii enemmän vaivaa, kuin yhden toteutuksen tekeminen, tai komponenttia ei käytettäisi usein, sen tuottaminen ei välttämättä ole sen tekemiseen kuluvan ajan arvoista. Myöskään, jos geneeristä toteutusta olisi aina räätälöitävä huomattavasti, ei uudelleenkäytettävä komponentti säästä aikaa.

Kehittäjien on keskenään sovittava, mitkä Drupalin ominaisuudet sopivat featuresien tuottamiseen. Eräitä tällaisia vaikuttavat olevan kentät, sisältötyypit, näkymät, sekä erilaiset asetukset kuten käyttäjäprofiilit ja käyttöoikeudet. Jos tuotettava komponentti kuuluu aiemmin mainittuihin, siitä voisi olla käytettäväksi perusominaisuuksia sisältävä features. Näin samoja perusasioita ei tarvitsisi tehdä uudelleen jokaisen projektin kohdalla.

Kehittäjien ongelmat joidenkin komponenttien kehitystyössä näyttäisivät johtuvan joidenkin konventioiden dokumentaation puutteista. Jos konventiot ymmärretään väärin tai kaikki kehittäjät eivät osaa hyödyntää niitä, konventioista ei ole hyötyä. Konventioista keskustelu, niiden kehittäminen tiiminä, sekä dokumentointi ovat oleellinen osa konventioiden käyttämistä osana ohjelmistoprojektia.

5.2 Kehitysehdotukset ja mahdolliset jatkotutkimukset

Tutkimuksen tärkeimmiksi kehityskohdiksi nousivat konventioiden dokumentointi ja niistä kertominen kaikille kehittäjille. Konventioiden dokumentaatio pitäisi olla aina kaikkien kehitystyötä tekevien saatavilla, jotta konvention noudattaminen onnistuu mahdollisimman helposti. Kun konventiosta löytyy dokumentaatio, tätä on myös muistettava ylläpitää muutoksen tapahtuessa. Lisäksi, jos konventioon tulee muutos, siitä pitää keskustella kaikkien kehittäjien kesken. Kun konventiot ovat kaikkien saatavilla ja tiedossa, niiden käytön ja toteutumisen varmentaminen onnistuu kaikilta tiimin jäseniltä.

Drupal-kehityksessä saattavat myös auttaa erilaiset ajatusmallit, kuten kyselyssä mainittu DRY. Kuten konventiot, myös ajatusmallit tulee esitellä kaikille kehittäjille. Kun kehittäjät työskentelevät projektien parissa samanlaisista lähtökohdista, heidän tekemiään valintoja esimerkiksi komponenttien rakenteessa voi olla helpompi ymmärtää. Ajatusmalleissa tärkeää ei ole niiden opettelu ulkoa, vaan kaikkien tiimin jäsenten yhteinen ymmärrys siitä, mihin kyseisellä ajatusmallilla pyritään.

Aihetta voisi tutkia lisää esimerkiksi tietyn ajanjakson jälkeen toteutettavalla kyselyllä. Kyselyssä voitaisiin käyttää jopa lähes samaa kyselypohjaa kuin tässä tutkimuksessa. Tuloksista voitaisiin silloin nähdä, ovatko kehittäjät hyödyntäneet aiemmin mainittuja kehitysehdotuksia työssään, ja millä tavoin, ja mitä hyötyjä tai haittoja tästä on mahdollisesti ollut.

5.3 Tutkimuksen luotettavuus

Tutkimuksessa tutkittiin vain hyvin pientä joukkoa, erään yrityksen Drupal-kehittäjiä ja heidän henkilökohtaisia tuntemuksiaan. Tämän takia tutkimustuloksia ei voida pitää yleispätevinä vastauksina, ja tutkimus toteutettiinkin tapaustutkimuksena. Tutkimustuloksien voidaan kuitenkin todeta tuovan arvoa yritykselle, jolle se toteutettiin, sillä siinä pyritään selvittämään voidaanko juuri kyseisessä yrityksessä tapahtuvaa kehitystyötä tehostaa.

5.4 Opinnäytetyöprosessin ja oman oppimisen arviointi

Opinnäytetyöprojektissa toteutettiin projektisuunnitelman mukaisesti tutkimus yrityksen Drupal-kehittäjien työn tueksi. Opinnäytetyöprojektin hankalin osa oli tutkimuskysymysten hahmottaminen tutkittavaksi kokonaisuudeksi. Myös projektisuunnitelmassa mainittu mahdollinen riski työhön käytettävän ajan puutteesta toteutui, joka johti työn valmistumispäivämäärän siirtämiseen useampaan kertaan.

Kun opinnäytetyön tutkimuskysymykset olivat selkeytyneet, työ eteni loppuun melko nopeasti. Opinnäytetyön aikana tapahtunut oppiminen liittyi suurimmalta osin Drupalin toiminnallisuuteen sekä erilaisten tutkimustapojen teorioihin. Projektinhallinnasta työ ei opettanut niin paljon kun projektisuunnitelmassa oltiin alun perin oletettu, vaikka kuukausia sitten tehdyn projektisuunnitelman muuttuminen sitä vastaavaksi valmiiksi opinnäytetyöksi olikin mielenkiintoinen prosessi.

Lähteet

Boiko B. 2005. Content Management Bible. Wiley Publishing, Inc. Yhdysvallat.

Butcher M., Dunlap G., Farina M., Garfield L., Rickard K. & Wilkins J.A. 2010. Drupal 7 Module Development. Packt Publishing, Iso-Britannia.

Chacon S. & Staub B. 2014. Pro Git. Apress. Yhdysvallat.

Crittenden M. 2014. Responsive Theming for Drupal. O'Reilly Media. Yhdysvallat.

Drupal.org. 2014. Working with CSS. Luettavissa: https://www.drupal.org/node/341246. Luettu: 15.3.2015.

Drupal.org 2015a. About. Luettavissa: https://www.drupal.org/about. Luettu 6.1.2015.

Drupal.org 2015b. Choosing a Drupal Version. Luettavissa: https://www.drupal.org/documentation/version-info. Luettu 29.4.2015.

Drupal.org 2015c. General Concepts. Luettavissa: https://www.drupal.org/node/19828. Luettu: 29.4.2015.

Drupal.org 2015d. Working With Blocks. Luettavissa: https://www.drupal.org/documentation/modules/block. Luettu: 29.1.2015.

Drupal.org 2015e. Building a Drupal site with Git. Luettavissa: https://www.drupal.org/node/803746. Luettu: 29.4.2015.

Drupal.org 2015f. Understanding the hook system for Drupal modules. Luettavissa: https://www.drupal.org/node/292. Luettu: 29.4.2015.

Drupal.org 2015g. Coding Standards. Luettavissa: http://www.drupal.org/coding-standards. Luettu: 21.2.2015.

Falk J. 2011. Drupal 7: The essentials. ImBridge NodeOne AB. Yhdysvallat.

Hodgdon J. 2012. Programmer's guide to Drupal. O'Reilly Media. Yhdysvallat.

Kumar K. 2012. Drupal 7 Theming Cookbook. Packt Publishing, Iso-Britannia.

Nixon R. 2014. Learning PHP, MySQL, JavaScript, CSS & HTML5. O'Reilly Media. Yhdysvallat.

PHP.net 2015. Preface. Luettavissa: http://php.net/manual/en/preface.php. Luettu: 21.2.2015.

Robbins J., Walker J., Eaton J., Haug N., Berry A. & Byron A. 2012. Using Drupal. O'Reilly Media. Yhdysvallat.

Tomlinson T. & VanDyk J.K. 2010. Pro Drupal 7 Development. Apress. Yhdysvallat.

Virtuaaliammattikorkeakoulu 2015. Induktiivisen päättelyn logiikka. Luettavissa: http://www2.amk.fi/digma.fi/www.amk.fi/opintojaksot/0709019/1193463890749/1193463919223/1193464257338/1193665352581.html. Luettu: 30.4.2015.

Liitteet

Liite 1. Kysely

Kysely yrityksen Drupal-kehittäjille

Komponentit: Moduulit (custom, contrib), Features, Teemat
Konventio: Yleinen tapa

1. Mitä komponentteja olet tuottanut viimeisen puolen vuoden aikana?
a) Kustomoituja moduuleita (Custom modules)
b) Features
c) Aliteemoja/teemoitusta (Subthemes, theming)

2. Minkälaisia komponentteja tuotat usein? (Esim. Feature samankaltaisista näkymistä)

3. Voiko tuottamiasi komponentteja käyttää uudelleen? Miksi kyllä tai ei?

4. Mitä konventioita käytät kehittäessäsi komponentteja?

5. Tukevatko käyttämäsi konventiot uudelleenkäytettävyyttä?

6. Onko olemassa konventioita, joita mielestäsi kannattaisi käyttää ja miksi?

Liite 2. Kyselyn kysymysten lajittelu analysointia varten tutkimuskysymysten ja teemojen perusteella

Tutkimuskysymys	Minkälaisia komponentteja tuotetaan usein?	Ovatko komponentit uudelleenkäytettäviä?	Onko käytössä konventioita?
Teema	Laatu	Uudelleenkäytettävyys	Konventiot
Kyselyn kysymykset	1. Mitä komponentteja olet tuottanut viimeisen puolen vuoden aikana? a) Kustomoituja moduuleita (Custom modules) b) Features c) Aliteemoja, teemoitusta (subthemes, theming)	3. Voiko tuottamiasi komponentteja käyttää uudelleen? Miksi kyllä tai ei?	4. Mitä konventioita käytät kehittäessäsi komponentteja?
	2. Minkälaisia komponentteja tuotat usein? (Esim. Feature samankaltaisista näkymistä)	5. Tukevatko käyttämäsi konventiot uudelleenkäytettävyyttä?	6. Onko olemassa konventioita, joita mielestäsi kannattaisi käyttää ja miksi?