HTML-ramverk i praktiken

En studie av HTML-ramverk från utvecklarens perspektiv

Johan Almgren
Rickard Andersson
Abstract

This paper aims to present a model for evaluating what we choose to call HTML-frameworks, a framework that contains a set of CSS layouts and occasionally JavaScript support, based on the characteristics of the software quality standard ISO/IEC 9126-1. The methods used to produce this model are based on a field study that involved making a web portal with the help of a HTML-framework, some relevant literature, and an analysis of the framework based on the characteristics of the ISO/IEC 9126-1 standard. The result was a model based on twelve different characteristics; functionality, web browser compatibility, collaborative skills, user-friendliness, documentation, visuals, graphic layout, performance, activity, professional support, replaceability, and license. This model should be viewed as a suggestion to what a model for choosing HTML-framework could look like, because we haven’t had the time to test it in a real situation, so further studies are needed, and we believe there could be improvements made in the objectivity of the assessments of the characteristics.
Innehållsförteckning

1. Inledning .............................................................................................................................. 5
   1.1 Bakgrund ......................................................................................................................... 5
   1.2 Syfte ................................................................................................................................. 6
   1.3 Frågeställning ................................................................................................................... 6
   1.4 Avgränsning ...................................................................................................................... 6
   1.5 Språkliga reservationer ..................................................................................................... 6

2. HTML-ramverk och relaterade begrepp ............................................................................... 7
   2.1 Ramverk ........................................................................................................................... 7
   2.2 Klient-server .................................................................................................................... 7
   2.3 Front-end & back-end ...................................................................................................... 8
   2.4 HTML .............................................................................................................................. 8
   2.5 CSS ................................................................................................................................. 8
   2.6 JavaScript ....................................................................................................................... 9
   2.7 HTML-ramverk ............................................................................................................... 9
   2.8 Webbramverk .................................................................................................................. 10

3. Metod ................................................................................................................................... 12
   3.1 Ämnesval ......................................................................................................................... 12
   3.2 Val av metod .................................................................................................................... 12
   3.3 Datainsamling ................................................................................................................. 12
      3.3.1 Fältstudie .................................................................................................................. 12
      3.3.2 Litteratur ................................................................................................................... 13
   3.4 Dataanalys ....................................................................................................................... 13
   3.5 Modellbyggande ............................................................................................................. 14
   3.6 Litteraturkritik ................................................................................................................ 14
   3.7 Metodkritik ...................................................................................................................... 14

4. Relaterad forskning ............................................................................................................. 15
   4.1 ISO/IEC 9126-1 ............................................................................................................. 15
      4.1.1 Functionality ............................................................................................................. 15
      4.1.2 Reliability ................................................................................................................. 16
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.20 Stability</td>
<td>29</td>
</tr>
<tr>
<td>5.21 Testability</td>
<td>29</td>
</tr>
<tr>
<td>5.22 Maintainability compliance</td>
<td>30</td>
</tr>
<tr>
<td>5.23 Adaptability</td>
<td>30</td>
</tr>
<tr>
<td>5.24 Installability</td>
<td>30</td>
</tr>
<tr>
<td>5.25 Co-existance</td>
<td>30</td>
</tr>
<tr>
<td>5.26 Replaceability</td>
<td>31</td>
</tr>
<tr>
<td>5.27 Portability compliance</td>
<td>31</td>
</tr>
<tr>
<td>5.28 Öppen källkod</td>
<td>31</td>
</tr>
<tr>
<td>5.29 Praktikbaserad kunskap</td>
<td>32</td>
</tr>
<tr>
<td>5.30 Software metrics</td>
<td>32</td>
</tr>
<tr>
<td>6. Modell för HTML-ramverk</td>
<td>34</td>
</tr>
<tr>
<td>7. Diskussion</td>
<td>37</td>
</tr>
<tr>
<td>8. Slutsatser</td>
<td>38</td>
</tr>
<tr>
<td>8.1 Vidare studier</td>
<td>38</td>
</tr>
<tr>
<td>9. Referenser</td>
<td>39</td>
</tr>
<tr>
<td>Bilaga 1, länkar till html- och webframverk</td>
<td>42</td>
</tr>
<tr>
<td>HTML-ramverk</td>
<td>42</td>
</tr>
<tr>
<td>Webframverk</td>
<td>42</td>
</tr>
</tbody>
</table>
1. Inledning

1.1 Bakgrund

Utvecklingen av webbaserade system håller idag allt högre och högre tempo, där systemen blivit allt mer komplexa vilket medfört att en snabbare utveckling bör ske för att hålla ned tidsåtgången. För att kunna följa med i det höga tempot behövs fler och bättre hjälpmedel, som underlättar utvecklingsarbetet. Ett sådant hjälpmedel är vad vi har valt att kalla HTML-ramverk (se en detaljerad förklaring under ”kapitel 2, HTML-ramverk och relaterade begrepp”). Ett ramverk innehåller många funktioner färdiga att använda som gör att det går snabbare att utveckla, men kan samtidigt vara väldigt begränsade på grund av sin strikta struktur, där utvecklarna måste göra korrekta referenser till HTML-ramverket för att det ska fungera.

Anledningen till att det går snabbare att utveckla webbaserade system med hjälp av ett HTML-ramverk är att det med ramverket följer fördefinierade formateringar för hur det grafiska utseendet ska vara för olika element, vilket gör att utvecklaren bara behöver referera sina element till dessa definitioner och slipper definiera egna värden för utseendet.

Det finns idag flera HTML-ramverk och det kan vara väldigt svårt att välja bland dessa, då alla har sina fö- och nackdelar. Att välja ett mindre lämpligt HTML-ramverk för sitt utvecklingsprojekt kan medföra att det kommer att öka utvecklingstiden då det kanske kommer att behövas att man bygger ut HTML-ramverkets funktioner för att passa projektet, då tappar HTML-ramverket lite av sin fördel då så mycket som möjligt redan ska vara fördefinierat i ramverket så utvecklaren bara behöver referera till ramverket och inte bygga ut dess funktionalitet. En annan fördel som är viktig att lyfta fram vid användandet av HTML-ramverk är att det medför att utvecklaren inte behöver besitta kunskap om CSS och JavaScript programmering i lika stor utsträckning, då dokumentationen för ramverket bör visa hur man använder det, så länge man inte behöver utöka HTML-ramverket funktionalitet det vill säga.

Om man använder sig av ett HTML-ramverk som använder sig av en layout som är allmänt utbredd bland webbplatser, till exempel att ha en fast bredd på sidan, med menyn för att navigera antigen till vänster eller högst upp, kan detta medföra att användaren kommer att känna igen layouten och lätta förstår hur man navigerar på webbplatsen. En risk vi kan se ifall många utvecklare använder samma ramverk är dock att alla system ser för lika ut och kreativiteten kring skapandet av användargränssnitt blir mindre, och nya kreativa lösningar kanske går förlorade.


1.2 Syfte
Syftet med denna uppsats är att ta fram en vetenskapligt grundad modell för att bedöma och välja HTML-ramverk. Denna modell ska kunna användas som en vägledning för utvecklare och projektgrupper när det ska beslutas om något HTML-ramverk ska användas och i så fall vilket som är det mest lämpliga ramverket att använda sig av.

1.3 Frågeställning
Vilka faktorer är nödvändiga att beakta vid valet av ett HTML-ramverk?

1.4 Avgränsning

1.5 Språkliga reservationer
2. HTML-ramverk och relaterade begrepp

Nedan följer några förklaringar på begrepp som kommer att användas i dokumentet för att underlätta läsningen.

2.1 Ramverk


2.2 Klient-server


En servers uppgift är att tillhandahålla resurser till de klienter som behöver dem, det kan vara hårdvara (till exempel en nätverksskrivare eller processorkraft), eller mjukvara (till exempel en e-post-tjänst eller en websserver som tillhandahåller webbsidor). En server är oftast passiv och ligger och väntar på anslutningar från klienter och aktiveras först när en klient skickar en begäran om en resurs.

Klienten är den datorn och det program som konsumerar resurser från en server, till exempel en webbläsaren som hämtar information från en websserver. Det är oftast klientens uppgift att initiera anslutningen till servern och begära dess resurser.

Ett exempel när det inte är klienten som ansluter till servern utan tvärtom är push-tekniken, som används bland annat vid “instant messaging” kommunikation över internet (Shuang & Kai, 2013), vilket är en textbaserad meddelandetjänst för korta meddelanden, där servern skickar ut inkommande meddelanden till klienten.
2.3 Front-end & back-end

Front-end är det grafiska användargränssnitt som användaren ser på klientsidan, utifrån ett webbperspektiv är det webbsidan i webbläsaren, detta gränssnitt kommunikerar i sin tur på ett förutbestämt sätt med ett back-end, återigen utifrån ett webbperspektiv är det webbservern. Front-end är ansvarig för att samla information från användaren och skicka det på rätt sätt så att back-end kan förstå det, till exempel om användaren vill söka efter någon form av information, back-end använder sig av informationen från front-end och utför sökningen på en eller flera underliggande källor och leverera slutligen ett svar till front-end som är ansvarig för att presentera det för användaren på ett begripligt sätt (“Front and back ends,” 2014).


2.4 HTML


```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css">
    <script src="javascript.js"></script>
    <title>HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

Figur 1. Exempel på HTML som skriver ut Hello World! i webbläsaren.

2.5 CSS

Cascading Style Sheets (CSS) är ett språk som är tänkt för att förenkla design och utveckling av webbsidor. CSS kombineras med märkspråk som HTML och XHTML, dessa märkspråk beskriver innehållet som ses i ett dokument; som länkar, rubriker, listor och tabeller. CSS däremot beskriver hur dokumentet och dess innehåll ska se ut (York, 2007).

CSS använder sig av selektorer som pekar ut vilket/vilka HTML-element som ska bli påverkade av CSS-koden (York, 2007). Det finns många olika selektorer som används när

I exemplet i figur 2 ser vi en CSS-kod där CSS-selektoren är en typ-selektor som pekar på alla HTML-element av typen <p>.

```css
p {
    text-align: center;
    color: red;
}
```

Figur 2. Exempel på CSS som ändrar utseendet och justeringen av texten “Hello World!” i exemplet från figur 1.

### 2.6 JavaScript

JavaScript är ett interpreterande programsspråk med stöd för objekt-oriente rad programmering. JavaScript används vanligtvis i webbläsare, och i den kontexten används JavaScript generellt för att tillåta skript att interagera med användare, ändra innehållet i dokument som visas inom webbläsarens fönster samt att kontrollera webbläsaren. Denna inbäddade version av JavaScript exekverar skript inbäddade inom HTML-webbsidor, det vill säga inbäddad JavaScript exekveras på klientsidan (Flanagan, 2006).

```javascript
window.onload = function () {
    document.getElementsByTagName("p").item(0).onclick = function () {
        document.getElementsByTagName("p").item(0).innerHTML = "";
    };
};
```

Figur 3. Exempel på JavaScript som tar bort texten “Hello World!” från exemplet i figur 1 när man klickar på texten.

### 2.7 HTML-ramverk


---

1 Programkoden tolkas samtidigt som programmet körs


Exempel på några HTML-ramverk är: 960 Grid System, Foundation, Bootstrap, Skeleton och HTML KickStart (Se Bilaga 1).

2.8 Webbramverk

Webbramverk är en samling verktyg som används för att underlätta utvecklingen av en applikation som exekveras på en server och levererar sin grafiska del till användarens webbläsare (Vosloo & Kourie, 2008). Webbramverk är med andra ord ett back-end ramverk som utför beräkningarna på serverdelen, men det kan också innehålla delar av front-end för att underlätta den visuella utvecklingen av en webbsida. Efter att webbramverket har gjort sina beräkningar genererar det oftast någon form av information som ska skickas till användaren, men informationen innehåller ofta inte någon formatering eller layout, utan webbramverket måste stoppa in informationen i HTML-kod. När det är gjort tar webbservern över och levererar den färdiga HTML-koden som webbramverket genererat, tillsammans med eventuella HTML-
ramverk och andra CSS- och JavaScript-filer, till webbläsaren som sedan ansvarar för att presentera innehållet för slutanvändaren.

Figur 5. Exempel på hur ett webbframverk kan hämta information från någon underliggande källa och därefter baka in den i HTML-kod.

Exempel på några webbframverk är: ASP.NET, PHP, JSP, Ruby on Rails (Se bilaga 1).
3. Metod

I detta kapitel kommer vi att gå igenom anledningen till varför vi gjort det ämnesval vi gjort, vilken metod vi valt, hur datainsamlingen gått till samt en beskrivning av hur dataanalysen och modellbyggandet har gjorts.

3.1 Ämnesval

Anledningen till att vi har gjort detta ämnesval beror delvis på ett personligt intresse för utveckling av webbaserade system och delvis på grund av att vi blev erbjudna ett projektarbete av ett företag i Skellefteå, vilket vi ville utföra. Detta projektarbete ledde, med vägledning av vår handledare på institutionen för informatik, till det aktuella ämnet HTML-ramverk.

3.2 Val av metod

“Kvalitativa studier bygger på en forskningsstrategi där tonvikten oftare ligger på ord än på kvantifiering vid insamling och analys av data” (Bryman & Nilsson, 2011, s 340)


3.3 Datainsamling

Datainsamlingen har genomförts genom en fältstudie samt genom litteraturläsning.

3.3.1 Fältstudie

Vi har genomfört en fältstudie hos ett företag i Skellefteå, vi kan kalla det företag X, där vi genomförde studien som fullvärda deltagare i ett projekt för att skapa en webbportal. Webbportalen uppgift var att presentera information från ett befintligt ärendehanteringssystem så att slutanvändarna av portalen kunde utföra grundläggande arbetsuppgifter inom ärendehanteringssystemet. Informationen hämtades, uppdaterades och togs bort via det existerande systemets API. Kraven som fanns från beställaren var att back-end skulle vara ASP.NET och C#, front-end var dock upp till projektgruppen att besluta om själva. Utöver kravet om back-end fanns det även ett antal krav om vilka funktioner webbportalen skulle ha, och hur den skulle fungera, men utseendet var helt upp till oss i projektgruppen. Projektet var
tidsbegränsat och vi hade drygt en månad på oss att leverera en första fungerande betaversion till beställaren. För att snabba på designen av webbportalen beslutade projektgruppen sig ganska tidigt för att använda HTML KickStart (se bilaga 1) som front-end HTML-ramverk, detta beslut fattades med hjälp av rekommendationer från en artikel som vi hittade genom en enkel sökning på nätet, vi använde oss med andra ord inte av någon modell för att välja ramverk.

Under arbetets gång har vi införskaffat en bred erfarenhet hur det är att jobba med ett HTML-ramverk.

Anledningen till att vi valde att göra en fältstudie var att vi ville prova att jobba med ett HTML-ramverk i en konkret situation, och genom detta skaffa oss erfarenhet kring hur ett HTML-ramverk var uppbyggt och hur man använder ett sådant.


3.3.2 Litteratur

De vetenskapliga artiklar vi använt oss av har delvis sökts fram genom sökning på Umeå universitets primo databas, med filtrering på vetenskapliga artiklar samt via ACM som finns länkat från Umeå universitetsbibliotek.

3.4 Dataanalys
även kompletterat analysen av HTML-ramverk med information kring öppen källkod, software metrics samt kunskap från praktiker via populärvetenskapliga artiklar.

3.5 Modellbyggande


3.6 Litteraturkritik

Det har varit väldigt svårt att hitta någon relevant litteratur för just HTML-ramverk, vi har däremot stött på en del litteratur inom området för webbramverk. Detta är anledningen till att stor del av uppsatsen kommer vara baserad på vår fältstudie, med stöd från den litteratur vi hittat. Den litteratur vi funnit behandlar information vars syfte är att göra det lättare för användarna, genom att standardisera och förbättra kvalitén på tekniker som används på Internet. Då programvarukvalité är ett ämne som tillhör informatik medför det att litteratur som försöker förbättra kvalitén på dessa tekniker också tillhör informatik området.

3.7 Metodkritik

Valet av att göra en fältstudie visade sig väldigt tidskrävande, då projektet vi genomförde hos företag X varade i fem veckor, vilket motsvarar halva tiden vi hade till förfogande för uppsatsen, vilket medförde att vi på grund av tidsbrist inte kunde genomföra en litteraturstudie då detta hade medfört ett allt för omfattande arbete tidsmässigt.

Om vi hade baserat vår studie på en mer ingående analys av de olika karaktäristika vi kommit fram till, hade en kvantitativ studie gjort att vi kunnat skapa mer konkreta mätvärden för att utvärdera ramverken. Vi hade genom en sådan studie kunnat designa och utfört tester av varje karaktäristika, för att skapa olika mätvärden, vilket hade kunnat ge vår modell en högre grad av objektiv bedömning för dess olika karaktäristika.

Ett annat sätt vi kunde ha gjort studien på är att intervjuar personer som är kunniga i ämnet, och fråga dem vad de anser är viktigt att tänka på vid val av HTML-ramverk, problemet vi ser med ett sådant förfaringsätt är att vi inte riktigt vet vilka personer som kan vara kunniga inom området, i så fall skulle vi först ha behövt försöka ta reda på det, och sedan lyckas boka intervju med dem och genomföra intervjun. Vi beslutade att eftersom kunskapen vi söker kräver en väldigt specialinriktad kompetens så skulle risken kunna vara väldigt stor att vi inte skulle få tag i tillräckligt med personer för att genomföra tillräckligt många intervjuer för att få en ordentlig bredd inom området.
4. Relaterad forskning


Vi belyser också öppen källkod och pratikbaserad kunskap vilket vi anser vara viktiga beståndsdelar till HTML-ramverk. Slutligen tar vi upp ämnet software metrics.

4.1 ISO/IEC 9126-1


Denna studie kommer att endast behandla de karakteristika som behandlas i ISO/IEC 9126-1; dessa huvudkaraktäristika är functionality, reliability, usability, efficiency, maintainability samt portability. Dessa karakteristika innehåller 27 underattribut (Krutz & Fry, 2009).

4.1.1 Functionality


Functionality är indelat i fem olika underattribut; suitability, accuracy, interoperability, security och functionality compliance. Suitability handlar om mjukvaruproduktens förmåga att leverera ett lämpligt antal funktioner för en specifikad uppgift samt användarens syfte. Interoperability handlar om mjukvaruproduktens möjlighet att interagera med ett eller flera specifikade system. Accuracy behandlar kapaciteten hos mjukvaruprodukten att leverera korrekt eller överenskomna resultat eller effekter med korrekt nivå av precision. Security syftar till hur mjukvaruprodukten skyddar information och data så att obehöriga personer eller system inte kan läsa eller editera dem, och samt att behöriga personer och system inte nekas tillgång till dem. Functionality compliance handlar om hur mjukvaruprodukten förhåller sig till standarder,
konventioner eller regleringar i lagar och liknande stadgars relaterat till funktionalitet (ISO, 1999).

4.1.2 Reliability
Reliability är en uppsättning attribut som behandlar förmågan hos mjukvaran att upprätthålla en viss nivå av prestanda under förutbestämda villkor och under en förutbestämd tid (EAGLES, 1995).

Reliability är indelat i fyra olika underattribut; maturity, fault tolerance, recoverability samt reliability compliance. Maturity belyser mjukvaruproduktens kapacitet att undvika haveri som ett resultat av fel i mjukvaran. Fault tolerance handlar om mjukvaruproduktens kapacitet att upprätthålla en viss nivå av prestanda ifall mjukvarufel eller intrång sker. Recoverability är mjukvaruproduktens förmåga att återskapa en specifiserad nivå av prestanda och återfå de data som blivit påverkad vid ett mjukvarufel. Reliability compliance belyser mjukvaruproduktens förmåga att förhålla sig till standarder, konventioner och regelverk relaterat till reliabilitet (ISO, 1999).

4.1.3 Usability


4.1.4 Efficiency
Efficiency är en uppsättning attribut som behandlar relationen mellan mjukvarans prestandanivå kontra hur mycket resurser som används, under angivna förhållanden (EAGLES, 1995). En annan definition hittas i ISO/IEC definitionen; Efficiency handlar om mjukvarans möjlighet att förse lämpligt mycket prestanda i relation till hur mycket resurser som används, under angivna förhållanden (ISO, 1999).

Efficiency är indelat i tre olika underattribut; time behavior, resource behavior och efficiency compliance. Time behavior handlar om mjukvaruproduktens kapacitet att leverera lämpliga svar, inom lämpliga responstider och med lämplig genomströmnings hastighet när systemet
utför dess funktioner, under angivna förhållanden. Resource behavior behandlar mjukvarans förmåga att använda lämpligt många och korrektta typer av resurser när mjukvaran utför dess funktioner under angivna förhållanden. Efficiency compliance berör förmågan hos mjukvarupro dukten att förhålla sig till standarder och konventioner relaterat till effektivitet (ISO, 1999).

4.1.5 Maintainability


4.1.6 Portability
Portability är en uppsättning attribut som behandlar mjukvarans möjlighet att förflytta från en miljö till en annan (EAGLES, 1995).


4.2 Öppen källkod
Programvara med öppen källkod är fri att använda, ändra och dela för alla. Således kan programvara med öppen källkod skapas, underhållas, uppdateras och användas i flera olika projekt, och av flera olika utvecklare. För att programvara ska räknas som öppen källkod måste den vara licensierad under någon av de licenser som följer direktiven för öppen källkod. Open

4.2.1 Direktiv för öppen källkod

För att få kalla sin mjukvara “Open Source”, enligt OSIs definition, räcker det inte med att man bara gör källkoden till sin mjukvara tillgänglig, utan man måste använda sig av en licens som följer alla (i skrivande stund) tio direktiven för öppen källkod ("The Open Source Definition | Open Source Initiative," n.d.). På frågan om man får kalla sin mjukvara för “Open Source” även om den inte följer någon godkänd licens, svarar OSI att de inte vill att man ska göra det för att det kan vilseleda användare av mjukvaran. Så vi tolkar det som att de inte kan vida några åtgärder om någon skulle missbruka “Open Source”, men att de hoppas att ingen gör det. Några exempel, från direktiven för öppen källkod, och vad de innebär är:


3. Licensen får inte sätta restriktioner på annan mjukvara, som är en del av ett paket med den licensierade mjukvaran. Till exempel får inte licensen kräva att annan mjukvara som levereras tillsammans med den licensierade mjukvaran är av öppen källkod.

4.2.2 Licenser

4.3 Praktikbaserad kunskap

Vi har tidigare nämnt att vi haft problem med att hitta vetenskapligt artiklar som behandlar HTML-ramverk. Dock används HTML väldigt flitigt på Internet i form av webbsidor, och det finns en hel del information om HTML och HTML-ramverk på Internet. Det är väldigt svårt att uppskatta hur många webbsidor som använder sig av HTML-ramverk, men vill man ha exempel på några som gör det kan man prova titta på någon av ramverkens webbplatser, där kan de ha exempel på andra webbplatser som använder sig av dem, till exempel "960 Grid System" (se bilaga 1) har flera exempel på sin webbplats om andra webbplatser som använder det ramverket.

Eftersom HTML-ramverk används i praktiken finns det en del praktikbaserad kunskap inom området, till exempel i form av att utvecklare som har provat olika HTML-ramverk delar med sig av sina erfarenheter.

När vi har undersökt saken under arbetet med vår uppsats har vi upptäckt att det finns två vanligt förekommande sätt att sammanställa ovanstående praktikbaserade kunskap, dels genom populärvetenskapliga artiklar, och dels genom sida-vid-sida-jämförelser av olika HTML-ramverk.

4.3.1 Populärvetenskapliga artiklar

Populärvetenskapliga artiklar kan ge oss en bild av vad som anses vara viktiga bedömningskriterier vid valet av HTML-ramverk, dessa kriterier kan redan innefattas av ISO/IEC standarden men kan också ge en bredare bild av vad allmänheten bedömer som relevanta kriterier.


4.3.2 Jämförelser av HTML-ramverk

4.4 Software metrics

Det verkar finnas väldigt mycket att undersöka när det gäller software metrics, ämnet verkar innehålla flera modeller och teorier bakom dessa, aldeles för många för att vi ska ha tid att sätta oss in i dem alla till den här uppsatsen. Det vi kommer göra är en snabb bedömning om det finns några software metrics vi kan använda oss av för att förbättra vår modell.
5. Analys av HTML-ramverk

Vi börjar med att analysera HTML-ramverk utifrån de 27 underkategorier som hör till ISO 9126-1. Efter det fortsätter vi vår analys av HTML-ramverk utifrån informationen om öppen källkod och praktikbaserad kunskap, slutligen listar vi några software metrics. Vid varje karaktäristika försöker vi bedöma vad som är rimligt, bland annat genom att använda oss av våra erfarenhet från vår fältstudie, att ta med i vår modell för val av HTML-ramverk.

5.1 Suitability


Vi tror det kan vara väldigt svårt redan i planeringsstadiet att tänka ut exakt vilka funktioner man kommer att behöva, vilket gör denna karaktäristika väldigt svår att bedöma i förväg. Trots att man aldrig kan veta helt och hållet vilken funktionalitet som kan behövas, kanske man har en aning om ungefär vilken funktionalitet som kommer behövas, då kan man utgå i från den och bedöma den mot ramverkets funktionslista, samt försöka göra en bedömning av HTML-ramverkets övriga funktionalitet så att den inte blir en belastning genom att till exempel ändra på ett element som inte ska ändras. Slutför man också försöka ta med i bedömningen hur mycket arbete som krävs för att eventuellt förändra eller bygga ut ramverket.


HTML-ramverks funktioner är något man i högsta grad borde beakta när man väljer bland dem, och tar med i vår modell under rubriken funktionalitet, då det är själva kärnan med olika HTML-ramverk att få ytterligare funktionalitet.

5.2 Accuracy

Hur mycket kan man lita på att webbsidan kommer att se likadant ut i olika webbläsaren? Finns det någon information om att HTML-ramverket har prövats i flera olika webbläsare? Detta kan vara av betydelse om man använder sig av många blockelement (ett element som normalt börjar och avslutas med en ny rad) har vi fått erfara, under vårt arbete hos företag X, då olika webbläsare inte tolkar alla storleksangivelser exakt likadant, vi hade ett problem med höjden på
en horisontell lista som inte blev lika stor i alla webbläsare och resulterade i att ramen runt den försvann i vissa webbläsare men inte i andra. Ett annat exempel är nyare funktioner som inte hunnit bli standard eller väldigt nyligen blivit det, kanske inte är implementerade i alla webbläsare, ett sådant exempel vi stötte på är gradients i CSS3 ("CSS3 Gradients," n.d.).

Det kan därför vara av stor vikt om man räknar med att ha användare med flera olika webbläsare att välja ett HTML-ramverk som har testats grundligt i alla webbläsare. Vi väljer att ta med denna karaktäristika i vår modell under rubriken webbläsarkompatibilitet.

5.3 Interoperability


5.4 Security
All säkerhet bör tillhöra servern, då det i princip inte går att uppnå någon form av säkerhet mot att data har blivit manipulerad hos klienten, vilket leder till att det underliggande webbramverket får stå för säkerheten, och vi väljer att inte ta med den karaktäristika i vår modell. Enkelt uttryckt, lita aldrig på data från användaren (Li & Xue, 2014).

5.5 Functionality compliance
Ett HTML-ramverk hjälper till största delen till med utformning och utseende av en webbsida, och vad vi vet finns det inga lagar och förordningar som förhindrar ett visst utseende. Det är mer
innehållet i så fall som måste kontrolleras så inget olagligt skrivs, men den kontrollen är det inte ramverket som är ansvarig för. Detta blir alltså ingen karaktäristika som vi tar med i vår modell.

5.6 Maturity
Det här är ett attribut som mer passar in på webbläsaren än HTML-ramverket, då det är webbläsaren som inte borde krascha även om det skulle visa sig att det finns fel i HTML-ramverket. Om så är fallet borde bara webbläsaren ignorera felet och fortsätta som vanligt.

Vi antar att det finns en möjlighet att det skulle kunna vara något fel i till exempel något JavaScript som tillhör HTML-ramverket och på så sätt finns det en teoretisk möjlighet, dock anser vi den vara mycket liten, att webbsidan blir deformered på något sätt. Vi har dock inte någon erfarenhet av detta själva. Vi har däremot erfarenhet från vårt arbete hos företag X av att om det blir fel i något JavaScript så kan en del funktioner sluta fungera på sidan, men den visas fortfarande korrekt. I vårt fall så använder vi oss av egna JavaScript för att dölja respektive visa olika delar av webbsidan, och när någon råkat orsaka ett fel i det JavaScriptet medförde det att sidan laddades in och såg ut som vanligt, men när man tryckte på någon av knapparna för att ta fram ett dolt element fungerade det inte.

Vi väljer att inte ha med denna karaktäristika i vår modell.

5.7 Fault tolerance
Återigen är det webbläsaren som bestämmer, om slutanvändaren till exempel väljer att stänga av CSS och JavaScript i sin webbläsare är det inget HTML-ramverket kan göra åt saken. Självklart är det bra om ramverket fungerar felfritt, om det till exempel innehåller en sorteringsalgoritm att den fungerar. Men HTML-ramverket execkeras inte i sig utan det är webbläsaren som execkverar HTML-ramverket, och enligt vår erfarenhet från företag X har de flesta webbläsare en mycket hög feltolerans och kan för det mesta presentera webbsidan för användaren även om den innehåller flera fel. Men då är det inte HTML-ramverket i sig som har en feltolerans utan det är webbläsaren som har det, och att försöka göra en bedömning av en webbläsarens feltolerans anser vi vara mycket svårt, och långt utanför denna uppsats, då samma webbsida kan se olika ut i olika webbläsare, och det kan vara väldigt svårt att bedöma om det beror på olika feltolerans i de olika webbläsarna eller om webbläsarna bara tolkar sidan annorlunda. Vi väljer därför att inte ta med denna karaktäristika i vår modell.

5.8 Recoverability
innehåller enligt vår erfarenhet de flesta webbläsare funktioner för att ta bort alla tillstånd och ladda om allt. Vi tar därför inte med denna karakteristika i vår modell.

5.9 Reliability compliance
Återigen är det upp till webbläsaren att hålla en viss standard när det gäller pålitlighet, HTML-ramverkets jobb är att styra utseendet av webbsidan, sen är det helt upp till webbläsaren att hantera att faktiskt visa webbsidan för användaren. Om till exempel användaren använder en webbläsare som inte officiellt stöds enligt dokumentationen av HTML-ramverket, och ramverket skulle ha specificerat något som webbläsaren inte förstår, för att det exempelvis är en ny CSS eller JavaScript funktion som inte hunnit implementeras i webbläsaren än, så ska webbläsaren bara ignorera instruktionen från HTML-ramverket och inte på något sätt bli påverkad negativt av det.


En validering av CSS-filer utifrån W3C går till på så sätt att man anger webbadressen till sina CSS-filer på sidan http://jigsaw.w3.org/css-validator/, sidan kollar sedan igenom CSS-filerna mot den standard som är satt av W3C och meddelar ifall CSS-filerna följer standarden eller inte, och ifall det inte gör det visas vad som ej följer standarden.

W3C validering kan man ha i åtanke när man väljer HTML-ramverk men vi kommer inte att ta med detta som en karakteristika vår modell. Argumentet för att inte ha med W3C i vår modell är på grund av karakteristikan webbläsarkompabilitet som vi tagit upp tidigare i vår modell, med webbläsarkompabilitet bör man kunna utläsa vilka webbläsare som har testats mot HTML-ramverket och således bör ramverket fungera i de angivna webbläsarna. Baserat på våra erfarenheter av att jobba i ett projekt hos företag X, med väldigt lite tid till förfogande, har vi kommit fram till att det är viktigt att ramverket fungerar i alla webbläsare och ser bra ut, även att det följer angiven standard och inte fungerar i de webbläsare som inte hunnit implementera standarden än.

5.10 Understandability
förståelse måste skapas inom dokumentationen för ramverket och väljer därför att lägga understandability i vår modell som en del av en karaktäristika vi väljer att kalla dokumentation.

5.11 Learnability

Generellt som programmerare anser vi det vara bättre om det man jobbar med är så lätt som möjligt att lära sig, och att det är så lätt som möjligt att förstå hur de som skapat det man jobbar med tänkt att man ska använda det, så man använder det på “rätt sätt”. Så det kan vara bra att försöka göra en bedömning av HTML-ramverk, att det inte är för komplicerat för de som ska arbeta med det.

Denna karaktäristika är väldig lik understandability då de båda behandlar användarvänligheten av HTML-ramverket, vi väljer därför att inte skapa någon ny rubrik i vår modell utan tar med learnability under rubriken dokumentation.

5.12 Operability
Denna karaktäristika har vi redan behandlat under de två ovanstående karaktäristika, den ingår under det vi kallar dokumentation i vår modell, då den handlar om att ta kontroll över HTML-ramverket och använda det på rätt sätt, och för att kunna göra det gäller det att man förstår hur det fungerar och hur man ska använda det, vilket vi redan har behandlat under ”learnability” och ”understandability”.

5.13 Attractiveness

Denna karaktäristika kan tyckas väldigt lik den om ”suitability” då ett HTML-ramverks grunduppgift är att tillhandhålla ett utseende för webbsidor, men rubriken funktionalitet i vår
modell omfattar inte riktigt det subjektiva som beskrivits i denna karaktäristika. Ett exempel på en tabell kan vara utifrån funktionalitet, att man kan få olika bakgrundsfärger på udda och jämna rader i tabellen, men sen är det en annan fråga om man tycker att det ser bra ut eller inte, och det är vad vi menar med det subjektiva, och det anser vi behöver en egen rubrik i vår modell.

Vi väljer att ta med denna karaktäristika i vår modell under rubriken utseende.

5.14 Usability compliance

Ett HTML-ramverk kan innehålla stöd för att grafiskt lägga till element baserat på olika riktlinjer för layout, en sådan stilistisk riktlinje kan till exempel vara “960 grid system”.


Vid valet av HTML-ramverket kan det underlätta att kolla ifall ramverket stödjer den grafiska layout man vill implementera som design för slutprodukten i sitt projekt. Att det till exempel finns klasser för att lägga till element i olika kolumner baserat på 960 grid system, om man vill använda sig av den layouten.

Från denna karaktäristika tar vi med rubriken grafisk layout i vår modell, då vi tycker att den skiljer sig från utseende, utseende är mera färg och form, medan layout är hur man väljer att strukturerar sin websida.

5.15 Time behavior


Ett HTML-ramverk tillför websidan ytterligare komplexitet, i form av CSS-selektorer och/eller extra funktionalitet i form av JavaScript, vilket kan medföra att det i slutändan går långsammare att ladda sidan om den använder ett HTML-ramverk än om den inte gör det, till exempel om ett JavaScript som är en del av ramverket måste exekveras när sidan är inladdad och detta tar två extra sekunder att slutföra. Detta har vi fått erfara hos företag X, då HTML-ramverket vi använde länkade till några typsnitt hos Google, och under en av dagarna vi arbetade tog Google:s server av någon anledning några extra sekunder på sig att svara, vilket
resulterade att webbsidorna vi byggde blev flera sekunder långsammare. Dock kan det vara riktigt svårt att avgöra hur mycket långsammare webbsidan kommer bli när man försöker välja bland HTML-ramverk, och att testa sig fram till ett resultat kommer antagligen också vara väldigt tidskrävande. Men det skulle antagligen gå att göra genom att skapa olika testsidor som använder olika ramverk och försöka mäta hur lång tid det tar att ladda de olika testsidorna. Därför kan det vara värmt att tänka på effektiviteten när man väljer bland HTML-ramverk och väljer att ta med det som rubriken prestanda i vår modell.


5.16 Resource behavior
Det yttersta ansvaret för resursåtgången ligger hos webbläsaren, det är webbläsaren som är ansvarig om något fel uppstår i resurshanteringen för en webbsida, till exempel att något JavaScript börjar leva sitt eget liv och allokerar för mycket minne eller fastnar i en evighetsloop. Men utöver detta yttersta ansvar, att förhindra så att datorn inte kraschar så är det upp till HTML-ramverkets skapare, att skapa ett så stabilt ramverk som möjligt, som använder så lite resurser som möjligt. Det handlar i slutändan om att webbsidan ska laddas så fort som möjligt och kräva så lite processorkraft som möjligt, det vill säga ha så hög prestanda som möjligt, och det har vi redan behandlat under rubriken prestanda i vår modell.

Det kan vara värmt att notera att det i grund och botten inte är själva webbläsaren som renderar själva webbsidan, utan webbläsaren implementerar en renderingsmotor som sköter det jobbet, renderingsmotorn består sen av flera mindre delar, till exempel renderingsmotorn WebKit som används av många webbläsare, till exempel Safari, Chrome, Android browser med mera, så sköts rendering, layout, inladdning och parsning av webbsidor av WebCore och JavaScript sköts genom JavaScriptCore (Kim, Lee, Lee, & Ro, 2013).

5.17 Efficiency compliance
På Internet råder normen att desto fortare det går desto bättre är det, men förutom det finns det ingen effektivitets standard, i alla fall inte vad vi vet eller kan hitta, när det gäller att ladda in en webbsida, och principen att det ska gå så fort som möjligt har vi redan med i vår modell under rubriken prestanda, så vi väljer att inte skapa någon ytterligare rubrik av denna karaktäristika.

5.18 Analyzability
En sak att övervåga om man är beredd att felsöka HTML-ramverk är att undersöka om källkoden finns tillgänglig eller om det bara finns i en så kallad minifierad version som är skapad för att ta så liten plats som möjligt, men är väldigt svår att analysera för en människa (“Minification
Enligt våra erfarenheter, från företag X, förutsatt att källkoden finns tillgänglig, går det ganska bra att analysera och felsöka JavaScript-kod, då webbläsarens utvecklingsverktyg oftast specificera exakt på vilken rad i koden felet uppstått. Det kan vara lite svårare att reda ut ett felaktigt utseende på något element särskilt om projektet innehåller flera stora CSS-filer, men det kan man också få ganska bra hjälp med att analysera och felsöka i webbläsarens utvecklingsverktyg. Dock är det betydligt enklare att桄 HTML-ramverket är väl strukturerat så att det är lätt att hitta till exempel CSS selektorer som behandlar ett visst element, antingen genom att all CSS är uppdelad på flera filer där till exempel en fil innehåller all information om tabeller, eller om HTML-ramverket består av en stor CSS fil att det finns många kommentarer och att filen på så sätt är indelad i sektioner. Det sistnämnda var fallet för oss hos företag X, mycket av CSS-koden låg i samma fil men det var ändå ganska lätt att hitta i den med hjälp av alla kommentarer.

Något som kan vara mer aktuellt är att ta reda på hur ofta HTML-ramverket uppdateras, och om det verkar ha aktiva utvecklare, finns det till exempel ett aktivt forum där man kan ställa frågor? Eftersom meningen med att använda HTML-ramverket är att det ska vara enklare att utveckla webbsidor med hjälp av ramverket, än utan det. Om de som skapat HTML-ramverket kan fixa eventuella fel som projektgruppen hittar, så inte medarbetare inom projektet måste lägga ner tid på det, blir det ännu enklare i utvecklingssyfte. Vi väljer att ta med detta i vår modell under rubriken utvecklaraktivitet.

5.19 Changeability

En av finesserna med CSS är hur olika selektorer ska prioriteras, mer specifika selektorer prioriteras högre än mer generella (“CSS Cascading and Inheritance Level 3,” 2013), detta medför att det är väldigt enkelt att ändra utseendet på ett element som får sitt utseende från generella selektorer genom att skapa mera specifika selektorer med ett nytt utseende åt elementet. Om HTML-ramverket till exempel har definierat ett visst generellt utseende på alla tabeller, men utvecklaren skulle vilja att en av de tabellerna som finns på webbsidan ska se ut på ett annat sätt, kan utvecklaren ge sin egen tabell ett klassnamn och definiera i en egen CSS-fil en mer specifik klass-selektor, så att alla tabeller med det klassnamnet får ett annat utseende. Detta medför att det finns två sätt om man vill ändra på utseendet på ett element som definerats av HTML-ramverk, dels att modifiera själva ramverkets CSS-selektorer, och dels genom att utöka ramverket med egna selektorer som beskrivits ovan. Båda sätten har sina egna för- och nackdelar:

Fördelen med att modifiera själva HTML-ramverket är enligt vår erfarenhet från företag X att det går väldigt snabbt, om man vill modifiera ordnade listor är det bara att söka på det, hitta ramverkets kod, ändra vad man nu vill ändra och sen spara. Nackdelen med det vi just beskrivit är att om man vill uppdatera HTML-ramverket till senaste versionen så går alla ändringar man gjort själv i ramverkets filer förlorade.

Fördelen med att utöka HTML-ramverket med egna filer är att man kan uppdatera själva ramverket men behålla alla egna ändringar man gjort i sina egna filer. Detta är också något som
vi har erfarenhet av från företag X och är den metod vi föredrog inom vår projektgrupp. Nackdelen med att använda egna filer enligt vår erfarenhet är att dels tar det längre tid och dels skulle det kunna bli krångligare att felsöka, om till exempel en ändring man gjort för sin ordnade lista är mer specifik än ramverkets så den får prioritet, men den råkar vara för generell och ändrar på så sätt också utseendet på andra ordnade listor, som tidigare fungerat. I detta fall skulle felsökningen för den som upptäcker felet med de andra listorna kunna bli krångligare när den personen inte kan hitta något fel i ramverket, eftersom felet nu finns i egna CSS-filer.

Vi väljer dock att inte ta med denna karaktäristika i vår modell, då alla ramverk går att förändra, det gäller bara för projektgruppen att välja vilken förändringsstrategi man vill använda sig av. Vi har också redan behandlat att det kan vara olika svårt att lägga till ny funktionalitet i olika ramverk, under karaktäristikan “funktionalitet” i vår modell.

5.20 Stability

Dock lider alla HTML-ramverk av ovanstående problem, vilket medför att ovanstående är information som kan vara bra att känna till när man väljer HTML-ramverk, men är desto viktigare att tänka på när man programmerar och testar webbsidan, vilket gör att vi väljer att inte ta med denna karaktäristika i vår modell.

5.21 Testability
Att testa förändringar i HTML-ramverket görs enligt vår erfarenhet, från arbetet hos företag X, på samma sätt som test av en webbsida. I projektet finns det förhoppningsvis riktlinjer om vilka webbläsare sidan ska fungera med och således bör testas i, och då är det bara att testa en webbsida som använder ramverket och se så att förändringen fungerar. Om projektet använder någon form av test-ramverk som till exempel hjälper till att testa i flera webbläsare samtidigt är det bara att använda det när HTML-ramverket testas också, då HTML-ramverket i grunden är ett antal CSS- och JavaScript-filer som kan testat på samma sätt som övriga CSS och JavaScript.

Vi har inte upptäckt något speciellt stöd för att testa något av de HTML-ramverk vi undersökt, dock brukar det framgå vilka webbläsare ramverken är testade i, vi har redan en rubrik för detta i vår modell som vi kallar webbläsarkompatibilitet. Vi väljer därför att inte skapa någon ny rubrik i vår modell av denna karaktäristika.
5.22 Maintainability compliance
Om projektgruppen är intresserade av att betala för professionell support kan det också vara ett läge att kolla upp om det aktuella HTML-ramverket erbjuder det, ett exempel på ett ramverk som erbjuder företag support är Foundation (se bilaga 1). Vi väljer att ta med professionell support som en ny rubrik i vår modell.

5.23 Adaptability
HTML-ramverk är designade för att exekveras i en webbmiljö som tillhandahålls av webbservern, underliggande miljö som själva webbservern exekverar i spelar ingen roll för HTML-ramverket, så själva webbservern kan befinna sig i vilken miljö som helst, i vilket system som helst, i vilket operativsystem som helst, på vilken hårddvara som helst. För själva HTML-ramverket spelar det ingen roll.
Om det går att anpassa ett HTML-ramverk att exekvera utanför en webbmiljö har vi dock ingen erfarenhet av, om det går och i så fall vilka användningsområden det har är frågor som sträcker sig utanför denna uppsats.
Inget HTML-ramverk behöver alltså anpassas till någon speciell miljö då de alla redan är gjorda för att exekveras i en webbmiljö och fungerar där utan någon anpassning, vilket medför att denna karaktäristika inte ingår i vår modell.

5.24 Installability
Enligt vår erfarenhet från företag X behöver inte HTML-ramverk någon speciell installation, de överförs till en webbserver på samma sätt som en webbsida, och sen är det webbserverns jobb att leverera dem till webbläsaren när någon person besöker webbsidan. HTML-ramverk behöver inte heller någon bearbetning av webbservern, utan kan levereras som de är direkt till webbläsaren, till skillnad mot ett webbframverk som först måste exekvera på servern innan dess utdata kan skickas till klienten. Den här karaktäristika ingår därför inte i vår modell.

5.25 Co-existance
Eftersom HTML-ramverk inte behöver installeras utan bara behöver finnas i filsystemet går det, enligt vår erfarenhet när vi testade att konfigurera webbservern hos företag X, att exekvera hur
många olika HTML-ramverk och websidor som helst i samma system, så länge som systemet klarar av det. Dock om HTML-ramverket kan samarbeta med andra ramverk på samma websida är en fråga vi redan tagit upp under “interoperability”, och som en rubrik i vår modell med namnet “samarbetssömnåga”.

5.26 Replaceability

5.27 Portability compliance
Alla HTML-ramverk är, per definitionen att de inte behöver installeras, och således inte behöver modifiera något på värdsystemet, portabla ("Portable Application Conversion Technology", 2010). Det är bara att kopiera in dem tillsammans med övriga webbsidor till den webbservrar som ska användas. Det finns därför ingen anledning att ta med den här karaktäristika i vår modell.

5.28 Öppen källkod
Om man ska använda någon annans mjukvara, vilket är fallet med HTML-ramverk är det nödvändigt att undersöka om man får använda den, och inte bryter mot mjukvarans avtal, det är här licenser kommer in i bilden. Av vår erfarenhet har nästan all mjukvara som finns att hämta på Internet någon form av licens kopplad till sig, som talar om hur man får använda mjukvaran,
vi har till exempel flera gånger varit i kontakt med mjukvara som har olika licenser för privatpersoner och företag, där programmet i fråga kan vara gratis att använda för en privatperson men kosta pengar för ett företag. Om däremot mjukvaran det vill säga HTML-ramverket använder sig av en öppen källkod licens, är det fritt att använda sig av den i vilket projekt som helst, både för privatpersoner och företag. Men man måste se upp med vilken öppen källkod licens som används, då de alla enligt direktiven för öppen källkod tillåter fri användning av icke modifierad källkod, men vissa öppen källkod licenser säger att om man modifierar källkoden måste resultatet av den modifieringen också licensieras under öppen källkod.

Alla fem av de HTML-ramverk vi listar som exempel i bilaga 1 använder sig av en öppen källkod licens och får således användas fritt av alla. Men det är inte säkert att alla andra HTML-ramverk är öppna, så det gäller för den som väljer bland ramverken att ta med det i beräkningen när det ska fattas beslut. Vilket gör att vi väljer att ta med licens som en karaktäristika i vår modell.

5.29 Praktikbaserad kunskap

Om vi nu analyserar HTML-ramverk utifrån de karaktäristika som framkommit vid läsning av praktikbaserad kunskap kan vi se att en majoritet av dessa karaktäristika redan blivit behandlade i vår analys. Vi inser att vi förbisett inlärningskurvan och lättförståelighet, då vi under analysen av ISO/IEC 9126-1 angett att liknande egenskaper borde ligga under karaktäristikan dokumentation. Vi inser nu att detta borde vara en egen rubrik och väljer att rubricera denna karaktäristika som användarvänlighet. Installationsens hastighet tillhör också karaktäristikan användarvänlighet, då det som menas med installationens hastighet är hur mycket konfiguration som måste genomföras innan ramverket kan tas i bruk, det vill säga hur användarvänligt det är att installera. Användarvänligheten handlar således om vilken komplexitet ramverket har, om ramverket måste konfigureras och hur lång tid det tar att lära sig ramverket.

5.30 Software metrics

Vi har som nämnts tidigare inte satt oss in i detta ämne fullt ut, utan vi tänkte bara använda oss av några kvantitativa mått om vi upptäcker något som passar i vår modell.

Laddningstid är ett mått inom software metrics som går att använda i vår modell till karaktäristikan “prestanda”. Genom att mäta laddningstid kan man få ett mått att jämföra mellan olika HTML-ramverk för att se hur de står sig mot varandra.

Vi inser också att man kan ge sig in och räkna antalet funktioner som finns i varje HTML-ramverk men det är väldigt mycket jobb att göra det manuellt, kontra vilken nytta man får ut av det, om det inte skulle finnas någon automatiskt metod att göra det på som vi inte känner till. Vi anser att det är lättare att bara försöka undersöka vilken funktionalitet som finns, och att den överensstämmer med de krav man har.
6. Modell för HTML-ramverk

Vi kommer nedan presentera den modellen som vi tagit fram för att utvärdera HTML-ramverk. Denna modell är ett resultat av analysen av HTML-ramverk utifrån ISO/IEC 9126-1 standarden, och övrig litteratur samt utifrån de kunskaper vi har om HTML-ramverk från vår fältstudie hos företag X. Modellen består av totalt tolv karaktäristika.

<table>
<thead>
<tr>
<th>Karaktäristika</th>
<th>Beskrivning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webbläsarkompatibilitet</td>
<td>Vilka webbläsare ska det webbaserade systemet stödja? När den frågan är besvarad måste man kolla ifall HTML-ramverket är testat för att fungera i de specifiserade webbläsarna. Denna information bör finnas på ramverkets hemsida.</td>
</tr>
<tr>
<td>Samarbetsförmåga</td>
<td>Om projektet kommer att använda sig av flera HTML-ramverk bör man först kolla upp om dessa kan samexistera med varandra eller om det uppstår krockar mellan ramverken. Detta kan man försöka utläsa från dokumentationen eller helt enkelt skapa en testsida där man använder ramverken sida vid sida för att se om det uppstår några problem.</td>
</tr>
<tr>
<td>Dokumentation</td>
<td>Är dokumentationen för ramverket tillräckligt för att utvecklaren ska få den hjälp som behövs för användning? Finns det exempel som visar hur ramverket ska användas? Finns det en översikt över vilken funktionalitet som följer med ramverket? Denna information bör finnas på ramverkets hemsida.</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Grafisk Layout</td>
<td>Vilket stöd innehåller CSS-filerna för den grafiska layout som planerats att användas i projektet. Denna karaktäristika handlar således inte om det visuella utseendet hos element utan om möjligheten att placera element där man vill utifrån den grafiska layouten. Detta bör man finna information kring i dokumentationen.</td>
</tr>
<tr>
<td>Prestanda</td>
<td>Kommer det valda HTML-ramverket göra så att sidorna blir mycket långsammare att ladda? I så fall, är det inom gränsen för vad som anses vara rimligt? Detta skulle man kunna testa genom att skapa en testsida som kan använda olika ramverk och mäta laddningstider och jämföra de mellan olika ramverk.</td>
</tr>
<tr>
<td>Professionell Support</td>
<td>Är man villig att betala för bättre support, erbjuder det aktuella ramverket det? Vilken support kan man i så fall förvänta sig? Detta borde framgå på ramverkets hemsida.</td>
</tr>
<tr>
<td>Ersätta befintligt</td>
<td>Om användarna av ramverket har tänkt ersätta ett befintligt HTML-ramverk, hur mycket jobb är det då att byta ut det mot det här nya? Används det ungefärligt samma upplägg på ramverket eller kommer man vara tvungen att göra drastiska förändringar i koden för att få det nya ramverket att fungera? Detta bör man försöka utläsa från det gamla ramverket och det nya ramverkets dokumentation för att försöka se ifall bytet går att genomföra utan att behöva koda om allt för mycket.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Licens</td>
<td>Vilken licens har HTML-ramverket? Tillåter licensen att ramverkets används i det syfte det är tänkt, till exempel tillåter den användning i kommersiellt syfte ifall det är planen med webbprojektet? Tillåter licensen modifisering av källkoden? Vilken licens som HTML-ramverket är licensierat under borde framgå av ramverkets hemsida.</td>
</tr>
</tbody>
</table>
7. Diskussion


Med citatet ovan vill vi poängtera att vår modell inte är en komplett detaljerad bild av vilka aspekter som man bör beakta vid valet av ett HTML-ramverk, utan vår egen uppfattning om vilka aspekter som är viktiga.


Den modell vi presenterar tycker vi fyller ett syfte i bedömningsprocessen av HTML-ramverk, dock kan den behöva byggas ut ytterligare. Vi har som tidigare noterats inte hittat litteratur som behandlar just utvärderingen av HTML-ramverk, detta kan bero på att dessa ramverk går under många olika namn vilket gör det svårt att söka eller att det helt enkelt inte skrivits så många vetenskapliga artiklar som behandlar ämnet.

En fördel med modellen är att den inte är allt för komplicerad och detta medför att bedömningen inte kommer att vara allt för tidskrävande, vilket gör att den kan vara ett enkelt hjälpmedel att använda för utvärdering av HTML-ramverk inom projekt.

Paradoxalt nog är modellens okomplicerade användning även en brist, modellen har inte något objektivt system för att bedöma vissa karakteristika, detta medför att bedömningen kan se olika ut i olika projekt då bedömningen förlitar sig mycket på subjektiva bedömningar från projektdeltagarna.

Modellen är således tillämpbar vid ett projekt där bedömningen av HTML-ramverket går att utföra utifrån subjektiva bedömningar, vid ett mindre projekt. Det kan behövas en utbyggd modell som innehåller objektiva bedömningar om denna modell ska användas i ett större mer påkostat projekt.
8. Slutsatser


Vi tycker att vi besvarat vår frågeställning kring vilka faktorer som är nödvändiga att beakta vid valet av ett HTML-ramverk i vår modell, och har på så sätt fullföljt syftet med denna uppsats.

8.1 Vidare studier

Modellen vi presenterar är endast ett förslag och kan behöva byggas ut ytterligare, andra analysverktyg bör testas för att se om modellen går att utökas. Modellen har inte blivit prövd i en konkret utvecklingssituation, vidare studier bör testa ifall modellen fungerar tillfredsställande i ett konkret projekt.

Modellen saknar även en objektiv bedömning för vissa karaktäristika och vidare studier skulle kunna försöka att implementera objektivitet med hjälp av till exempel ytterligare software metrics, då detta inte hunnits med i denna uppsats på grund av den begränsade tid vi haft till vårt förfogande.
9. Referenser


Bilaga 1, länkar till html- och webbframverk

Exempel på några av de ramverk vi undersökt, samt vart på Internet man kan hitta dem.

**HTML-ramverk**

<table>
<thead>
<tr>
<th>Namn</th>
<th>Länk</th>
<th>Licens</th>
</tr>
</thead>
</table>

**Webframverk**

<table>
<thead>
<tr>
<th>Namn</th>
<th>Länk</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP.NET</td>
<td><a href="http://www.asp.net/">http://www.asp.net/</a></td>
</tr>
<tr>
<td>Ruby on Rails</td>
<td><a href="http://rubyonrails.org/">http://rubyonrails.org/</a></td>
</tr>
</tbody>
</table>