Göteborgs Universitet
Institutionen för informatik

Teknik för design och management av websiter

Fredrik Kraft och Jesper Wertsén

Abstrakt

Syftet med denna uppsats är att undersöka vilka tekniker och verktyg som är lämpliga att använda vid utveckling och management av interaktiva websiter. Uppsatsen består av en inventering av de verktyg och tekniker som finns idag. Därefter följer en analys och jämförelse mellan verktygen, med avseende på olika parametrar (plattformar, kostnader, underhåll, användarvänlighet, resursbelastning och grafiska möjligheter). De verktyg och tekniker som vi undersökte var: HTML, JavaScript, Java, CGI, Flash, XML, JSP och ASP. Det mest lämpliga verktyget för interaktiva internetlösningar var JavaScript, men även Flash och Java är relativt användbara.

Magisteruppsats 20 p, Fristående kurs
Höstterminen 1999

Handledare Informatik: Birgitta Ahlbom
Handledare Ericsson Microwave Systems: Mikael Henström

Innehållsförteckning

1 Inledning

1.1 Bakgrund

När vi diskuterade om vi skulle skriva en magisteruppsats kom vi fram till att vi ville utreda något med anknytning till internet/intranet. Anledningen till att vi ville studera just internet/intranetlösningar är att vi tycker att detta är ett av de mest intressanta områdena inom IT-branchen för att tekniken utvecklas fort och det är ett väldigt aktuellt ämne. Vi ställde upp två krav: först och främst skulle uppsatsen inte enbart omfatta teori, utan även innehålla en praktisk del. Vidare ville vi att examensarbetet skulle ha en praktisk anknytning till ett IT-företag.

Ericsson Microwave Systems (EMW) i Mölndal är ett företag som bland annat utvecklar flygplansradar. Det består av en civil och en militär del, som sedan i sin tur är uppdelat i flera olika divisioner.

EMW har tidigare haft ett stort antal verktyg för projektstyrning och projektuppföljning, men på senare tid har företaget tagit fram gemensamma verktyg och sållat bort en del gamla. De nya verktygen som införts är specialgjorda för EMW. Detta i sin tur innebär att de som skall använda verktygen inte har tillräcklig kunskap om hur de fungerar. Det finns inte heller några utförliga manualer angående verktygen. UGCC är den enhet som leder utvecklingsarbetet av de nya verktygen och enheten har fått väldigt många frågor om hur dessa skall användas. Detta ledde till en förfrågan från Mikael Henström, utvecklingsansvarig för några av dessa hjälpmedel på EMW, till Institutionen för informatik om det fanns någon som var intresserad av göra ett examensarbete som omfattade hjälp med manualerna och en site som strukturerar upp projektledningsverktygen. Vi tyckte detta lät spännande och kontaktade EMW. Efter ett inledande möte med Mikael Henström och Rolf Magnusson (ansvarig på UGCC) stod det klart för oss att manualerna skulle vara tillgängliga via EMWs intranet. För att kunna utveckla manualerna behövde vi först ta reda på vilka verktyg och tekniker som var mest lämpliga att använda för intranetutveckling. Detta låg helt i linje med våra önskemål om att skriva en uppsats med anknytning till internet och som också omfattar en praktisk del.

1.2 Problemformulering

Eftersom vi tycker att internettekniken är mycket intressant ville vi undersöka vilket/vilka verktyg eller teknik som är mest lämplig att använda för utveckling och management av en interaktiv website. För att ge svar på vilket/vilka verktyg som är mest lämpligt formulerade vi följande frågor.

Vilka tekniker och verktyg finns för utveckling av interaktiva internetlösningar?

Vilka verktyg/tekniker (VT) är lämpliga att använda sig av med avseende på nuvarande tekniska nivå i IT-samhället?

1.3 Avgränsningar

Följande områden kommer inte att behandlas i uppsatsen:

- PHP3 (skript som körs på webservern)
- Användandet av ljud i interaktiva internetlösningar
- Videotekniker för internet
- Mobila internetlösningar

1.4 Syfte

Syftet med denna uppsats är att undersöka vilka tekniker och verktyg som är lämpliga att använda vid utveckling och management av interaktiva websiter.

1.5 Rapportstruktur

Uppsatsen består av sex kapitel: inledning, inventering av VT, analys och jämförelse av olika VT, tolkning av resultat, metod och slutsatser.

I kapitlet ”Inledning” beskriver vi bakgrund, problemformulering, avgränsningar, syfte och rapportstruktur. I kapitlet ”Inventering av Verktyg och Tekniker (VT)” beskriver vi de olika VT för internetutveckling. Sedan jämför vi dessa i kapitlet ” Analys och jämförelse av olika VT” med avseende på sex parametrar. I detta kapitel presenterar vi även resultatet i en matris. I kapitlet ”Tolkning av resultat” tolkar vi resultatet. I kapitlet "Metod" beskriver vi tillvägagångssättet vi har använt i denna uppsats och presenterar alternativa metoder. I kapitlet ”Slutsater” presenterar vi våra slutsatser av uppsatsen, diskuterar vad vi kunde ha gjort annorlunda och ger vi förslag på fortsatt forskning.

2 Inventering av verktyg och tekniker (VT)

I detta kapitel beskriver vi olika VT för framställning av interaktiva websiter.

När vi har beskrivit de olika VT:n har vi hämtat information från en rad böcker och artiklar. Beskrivningarna är en sammanställning av informationen och vi har därför valt att ange källhänvisningarna under respektive rubrik, då det inte går att direkt hänvisa till den exakta källan vid olika tillfällen; på grund av redundant information.

2.1 HTML

(Lemay, 1995), (Solberg, 1997), (Ladd & O'Donnell, 1998), (Pfaffenberger & Gutzman, 1998), (*HTML Home Page*)

HTML står för Hyper Text Markup Language och är baserat på SGML (Standard Generalized Markup Language). SGML används till att beskriva den generella strukturen hos olika typer av dokument. HTML är inte ett sidbeskrivningsspråk som t.ex. PostScript. HTMLs fokus är på innehåll och inte utseende. Det är ett språk för att beskriva strukturerade dokument. Tanken bakom det hela är att de flesta dokument har gemensamma element såsom titel, stycken osv. Genom att skapa en uppsättning av de element som ett dokument kan innehålla, går det vid produktionen av dokumentet att på ett smidigt sett använda sig av de förutbestämda elementen för att få en homogen struktur. I HTML kallas dessa element för taggar (tags på engelska). Det är taggarna som beskriver dokumentet och allt som inte är en tagg är en del av själva dokumentet.

Det behövs en webläsare för att hämta dokument via internet. Detta är inte webläsarens enda funktion utan det finns flera andra. En av de mest fundamentala delarna i webläsaren är HTML-formateraren, som formaterar dokumentet. När ett dokument laddas, läser ("parsar") webläsaren HTML-informationen och formaterar sedan texter och bilder på skärmen. Formateringen skiljer sig mellan de olika webläsarna, vilket innebär att samma dokument kan se olika ut i olika läsare. Skillnaderna var dock mycket större mellan de tidiga versionerna av webläsare än dagens.

World Wide Web Consortsium (W3C) är en grupp akademiska och industriella partners som utvecklar HTML-standaren för World Wide Web. W3C drivs av Massachusetts Institute of Technologys (MIT) laboratorium för datorteknik i Massaschusetts, USA samt Institut National de Recherche en Informatique et en Automatique (INRIA), ett vetenskapsinstitut i Frankrike som ägnar sig åt forskning inom datorteknik.

Nya förslag till förändringar av reglerna för HTML kan lämnas av medlemmar i W3C eller professionella i datorbranschen till W3C. Om gruppen anser att det finns fog och möjlighet att införa en ny regel, inkluderas den nya regeln i nästa version av HTML-standarden. Den uppsättning regler för HTML som gäller vid en viss tidpunkt kallas för aktuell HTML-standard eller -version. Efter lanseringen av HTML som gjordes av Europeiska Centret för Energifysik i Schweiz (CERN) 1989 har det funnits en rad HTML-versioner.

2.1.1 Historik

HTML 0 (1989) Denna version gav endast stöd för enkla formateringsanvisningar som avsnittsmellanrum, rubriknivåer och listor.

HTML 1 (tidigt 90-tal) Detta är HTMLs första stora utvecklingssteg och drivkraften till det var önskan att införa grafik i dokumenten. Vid denna tiden lanserades Mosaic, en webläsare, som blev föregångare till dagens grafiska webläsare.

HTML 2.0 (1994) W3Cs version HTML 2.0 var ett försök att fastställa hur HTML användes vid den tidpunkten. Avsikten var också att fastställa alla grundläggande HTML-element för att få en skiljelinje mellan standard-HTML och webläsarbunden HTML. Speciellt Netscape försökte vidga gränserna genom att lansera många element som bara webläsaren Netscape Navigator kunde tolka på ett korrekt sätt. Till en början skapade detta givetvis tolkningsproblem hos webläsare från andra tillverkare.

HTML 3.0 (1995) HTML 3.0 baserades på en serie regelförslag till W3C som skulle utöka HTMLs funktionalitet och erbjuda stöd för tabeller, matematiska symboler med mera. Denna version antogs dock aldrig av W3C eftersom man inte kunde få en överenskommelse till stånd på alla punkter. Istället hoppade W3C till vad man kallade HTML 3.2. Detta var en utökad version av HTML 2.0 som innehöll många, men inte alla, av de förslag som gjorts till HTML 3.0

HTML 3.2 (maj 1996) Nyheterna i denna version var:
- stöd för tabeller
- attribut för att kontrollera bakgrunder, text och länkfärger
- bättre positioneringskontroll i block och för textelement
- utökat bildstöd, inklusive möjlighet att flöda text runt en bild
- utökade teckensnittsstilar
- stöd för flerlänkade bilder (image maps) även på klientsidan
- stöd för integrerade Javaapplets

HTML 4.0 (juli 1997) Den nya standarden, som gäller idag, innehöll tillägg inom bland annat följande områden:

- stöd för flera fönster (frames)
- stilformatmallar (style sheets)
- möjlighet att inkludera skript
- stöd för matematiska uttryck, tecken samt förbättrande rutiner för utskriftshantering, mer avancerade formulär och bättre tabellkontroll

2.1.2 Tekniska plattformar

Den senaste versionen av HTML (4.0) stöds av Internet Explorer 4.0 och senare versioner. Netscape Navigator 4.0 och senare versioner stöder dock inte standarden fullt ut.
Eftersom HTML bara använder de grundläggande funktionerna hos webläsarna behövs inga ytterligare tilläggsprogram (plugins).
De nya versionerna av HTML är gjorda så att dokument skrivna i äldre versioner kan läsas av webläsare som stöder de nya versionerna. Varje HTML dokument är litet, vilket gör att det kan laddas väldigt fort över nätet. Dokumentet innehåller dock varken ljud eller bild, utan endast hänvisningar (länkar) till dessa. Då ljud och bildfiler är relativt stora leder detta till att hemsidor som innehåller dessa effekter tar ganska lång tid att ladda in. HTML är gratis och vem som helst får använda standarden. HTML har den fördelen att det går att skriva den i nästan vilken editor som helst.

2.1.3 Underhåll

Fördelen med HTML-dokument är att det uppdaterade dokumentet inte behöver bearbetas ytterligare (kompileras om) innan vilken webläsare som helst kan läsa det. En nackdel vid uppdatering av HTML-dokument är att koden tenderar att bli ganska rörig i omfattande dokument.

2.1.4 Användarvänlighet

HTML innehåller ett begränsat antal element (taggar) som är lätta att lära sig. I och med att det endast går att använda fördefinierade taggar och det inte går att skapa några nya funktioner, begränsas möjligheterna till interaktion och ny funktionalitet. En stor fördel med HTML är att det finns nästan obegränsat med bra litteratur i både tryckt och elektronisk form.

2.2 JavaScript

(Goodman, 1998)

JavaScript är ett programmeringsspråk som används integrerat i HTML-dokument. JavaScript har, till skillnad från Java, liten syntax och en enkel programmeringsmodell. Det är fler saker som skiljer de olika språken åt. De har t.ex. helt skilda programtolkar för att köra kod och olika stöd hos de olika webläsarna. Java i webformat (applet) är kompilerat, vilket JavaScript inte är eftersom det är ett skriptspråk och tolkas direkt i webläsaren.

Det finns en mängd lösningar som kan skapas med hjälp av JavaScript. JavaScript är användbart när t.ex. formulärelement (bl.a. textrutor, textareor, knappar, radioknappar, checkboxar, selektionslistor) skall påverkas direkt av det användaren gör. Det kan vara att en meddelanderuta dyker upp när ett felaktigt värde fyllts i en textruta eller att ett val i en selektionslista tar användaren direkt till den länk som valts utan att man behöver klicka på några fler knappar.

Andra situationer då JavaScript är lämpligt är då mindre mängder data skall lagras lättåtkomligt och med snabb accesstid i något som liknar en databas. Navigering mellan ramar, plugins parametrar och Javaapplets kan enkelt styras av användaren via JavaScript. Ibland kan data behöva tolkas innan den vidarebefordras till en server, då är återigen JavaScript ett bra val.

För att skydda den personliga integriteten har det byggts in en rad begränsningar i JavaScript. Ett exempel på dessa begränsningar är att det inte skall gå att konstruera hemsidor som ”rotar” på surfarnas datorer. Några av de saker som inte går att göra med hjälp av JavaScript är:

- Påverka eller hämta in information om webläsarnas inställningar, fönsterinställningar, ”action”-knappar och utskrifter.
- Starta program på klientdatorn.
- Läsa eller skriva till filer eller kataloger på klientdatorn.
- Läsa text på en HTML-sida eller andra filer som ligger på servern.
- Skriva filer till servern.
- Läsa mappar på servern.
- Fånga upp ”levande” dataströmmar från servern.
- Skicka hemliga e-post meddelanden från en websida.

2.2.1 Historik

Under 1995 började Netscape utveckla ett skriptspråk som kallades LiveScript. Från början skulle detta språk kunna användas för två olika ändamål med samma syntax. På

serversidan skulle det användas till att administrera webservrar och kopplingar till bland annat ”back-end” databaser och sökmotorer. På klientsidan skulle skripten förbättra websidorna på flera olika sätt som t.ex valideringskontroll vid inmatningar i formulär (för att slippa onödig belastning av servern) och för beräkningar som inte tvunget måste göras på servern. Senare kom språket också att användas till att förmedla information mellan HTML-dokument och Java-appletar.

Netscape och Sun (de företag som skapade Java) bytte i början av december 1995 namn från LiveScript till JavaScript. Namnbytet ledde till en viss förvirring och JavaScript jämställdes felaktigt i många fall med Java. Sanningen var att JavaScript bara hade lånat en del grundläggande syntax från Javan, som i sin tur hade ärvt dessa från C/C++.

I februari 1996 kom Netscape 2.0 med stöd för JavaScript 1.0 för alla plattformar. Stödet för Java var mer begränsat i samma version av samma webläsare, t.ex. fick inte Windows (3.1) användare tillgång till Java.

Netscape Navigator 3.0 stödde funktioner som gjorde att JavaScript (1.1) och Java kunde användas smidigare tillsammans.

De första versionerna av Internet Explorer (IE) hade inget stöd för varken JavaScript eller Java, men i och med IE 3.0 så stöddes båda. IE 3.0 låg dock efter en del i utvecklingen. Netscape 3.0 stödde JavaScript 1.1 och IE 3.0 bara version 1.0 (eller JScript som den kallas i Microsofts version). De senare versionerna av IE har förutom standard JavaScript haft egna specialvarianter. Netscape Navigators senare versioner har även de haft egna specialtillägg.

JavaScript (1.2) stöds från och med version 4 av både Netscape Navigator och IE.

2.2.2 Tekniska plattformar

JavaScript 1.2 stöds av både IE och Netscape Navigator från version 4 och uppåt på alla plattformar. Denna version och de tidigare versionerna av JavaScript är bakåtkompatibla. JavaScript kräver inte mycket av klientdatorn och är gratis eftersom det ingår i webläsarna.

2.2.3 Underhåll

JavaScript är relativt enkelt att underhålla eftersom de allra flesta HTML-editorerna har stöd för språket och debuggingverktyg inbyggda. För att uppdatera JavaScript-kod

behövs ingen kompilator och därför räcker det oftast att webläsarna laddas om för att ändringarna skall verka.

2.2.4 Användarvänlighet

Språket JavaScript är lite svårare att bemästra jämfört med HTML eftersom det finns fler kommandon etc. Det är lite problematiskt att arbeta med JavaScript i de äldre läsarna (IE 3 och Navigator 2-3) eftersom det är stora inkompabiliteter mellan läsarna. Det finns ett stort utbud av onlinehjälp, nerladdningsbara kodsnuttar och vanliga programmeringsböcker om JavaScript.

2.3 Java

(Bishop, 1997), (Rice & Salisbury, 1997), (Ladd & O'Donnell, 1998), (*The Java Tutorial*), (Flanagan, 1997)

Java är ett generellt programmeringsspråk som kan användas för alla typer av applikationer. Språket har influerats av många andra språk såsom Smalltalk, Fortran och C/C++. När ett Javaprogram kompileras genereras s.k. bytekod som är ett standardiserat instruktionsformat för en virtuell Javamaskin. Denna virtuella Javamaskin är den som gör Java till ett plattformsoberoende språk. En virtuell Javamaskin är ett program som är skrivet för en viss plattform (Windows 95, Unix, Macintosh etc.) och som först tolkar bytekod för att sedan utföra instruktionerna i denna. I och med att bytekodens format är standardiserat kan alltså en fil med bytekod exekveras på samtliga plattformar som har en virtuell Javamaskin, och dessutom bete sig på samma sett oavsett plattform. Det är alltså samma bytekod som exekveras men olika virtuella maskiner används på olika plattformar.

Java används över Internet genom att den kompilerade och kompakta bytekoden läses ned från en webserver och exekveras i en webläsare med en inbyggd virtuell Javamaskin. Den virtuella javamaskinen i webläsaren är specialskriven för samma operativsystem som webläsaren.

De egenskaper som kan tillskrivas programmeringsspråket Java är att det är:

- Enkelt: Java anses betydligt enklare att lära sig än andra språk, som exempelvis C++ och Smalltalk. Trots detta är språket komplett, inga viktiga konstruktioner saknas utan snarare är det fällorna i C++ som har tagits bort.
- Objektorienterat: Java är fullt ut objektorienterat i och med att det har stöd för klasser, objekt, arv och polymorfism.
- Plattformsoberoende: Genom införandet av en virtuell maskin som läser den oberoende bytekode är Java också plattformsoberoende. Med plattformsoberoende

menas både oberoende av en viss processor, ett visst operativsystem och ett visst användargränssnitt. Detta är en egenskap som få andra språk har och som i Java dessutom skapas på binär nivå, dvs källkoden behöver inte kompileras om för olika plattformar utan bytekoden kan direkt förflyttas mellan olika maskiner.

- Hårt typat: Java är skapat med filosofin, hårda kontroller vid kompileringen så att fel upptäcks redan då och inte under exekvering av ett program. Genom att svårhanterliga pekare inte finns med i språket undviks många av de typproblem som är vanliga i andra språk.
- Robust: Java underlättar skapandet av robusta program, dvs program som är förutsägbara och som inte kraschar. Även om det slutgiltiga ansvaret för att uppnå robusthet är upp till programmeraren att se till, innebär avsaknad av pekare, automatisk minneshantering och stöd för undantagshantering att programmeraren får det betydligt enklare att utveckla robusta program.
- Trådat: Många operativsystem har numera stöd för parallell exekvering genom sk. trådar (threads) där ett program kan innehålla många samtidiga exekveringsvägar. Normalt är dessa mekanismer svåra att programmera, men inte om Java används. Java har inbyggt stöd i språket både för att skapa och styra trådar, samt eleganta mekanismer för att synkronisera trådar med varandra.
- Säkerhet: Med säkerhet kan man också mena robusthet som redan diskuterats, men eftersom Java används över Internet så menar man med säkerhet oftast skydd mot missbruk av Java. Detta för att inte någon skall kunna skriva Javaprogram som överförs via Internet och sedan förstör eller tar reda på information på en annan dator. Java har därför en välutvecklad säkerhetsmodell för att låta okända program exekvera under stark kontroll och med begränsade rättigheter. Även om det aldrig går att säga att någonting är 100% säkert, så är det dock ganska säkert att Java-tekniken med en virtuell maskin som implementerar en väldefinierad säkerhetsmodell är betydligt säkrare än många andra konkurrerande tekniker (t.ex. Active X) för att överföra program över Internet.
- Standardiserat: Java är redan idag ett mycket väldefinierat språk som inte lämnar mycket tolkning kvar för kompilatortillverkare. Detta har åstadkommits genom att både språket, tillhörande klassbibliotek samt den virtuella Java-maskinen har specificerats i detalj. Trots att Java redan idag alltså är väldefinierat, pågår också arbete med att skapa en officiell och internationell standard.
- Distribuerat: Java är genom sitt Internet-stöd distribuerat genom att den kompilerade bytekoden läses ned från en server och exekveras på en klient. I de standardiserade klassbiblioteken finns dock ytterligare stöd för att skriva mer avancerade klient-server system där logik och objekt fördelas på klient och server, och där objekt kommunicerar med varandra över nätet och kod transporteras dynamiskt över nätet allteftersom behov av koden uppstår.

Ett Javaprogram kan finnas i två olika former: applikationer och applets. En Java-applikation är ett program som exekveras fristående från andra program. En Java-applikation beter sig som vilket annat program som helst och använder sig av Java

API:erna (Advanced Programming Interface) för att komma åt tjänster från operativsystemet.
En Java applet är ett program som alltid exekverar inifrån ett program, normalt en webläsare. Java applets är formen för att distribuera Javaprogram över Internet eller ett intranet.

Vad händer då när en Java applet exekveras via nätet? En applet ligger inbäddad som en del av en HTML-sida. Ett speciellt HTML-kommando används som refererar till den fil där appletens bytekod ligger. Filen med appletens bytekod läses ned tillsammans med alla andra filer (grafikfiler, ljudfiler etc) som sidan använder sig av. Appleten tilldelas ett utrymme i form av en viss area på den fysiska sida som skall visa HTML-sidan. Bytekoden för sidan börjar sedan exekveras i den virtuella Javamaskin som ligger innesluten i webläsaren, och denna exekvering påverkar innehållet i det tilldelade utrymmet på sidan. En applet startas på ett litet annorlunda sätt än en Java-applikation.

Det är inte möjligt att exekvera en applet som en applikation eller vice versa. Då en applet laddas ned över nätet har den också större säkerhetsrestriktioner inbyggda. Det finns ett antal saker som en applet inte tillåts göra på klientmaskinen som t.ex. att komma åt hårddisken på maskinen.

Hittills har innehållet på websidor varit i huvudsak statiskt, d.v.s. innehållet har varit text och bild. Protokollet för detta innehåll har dock varit standardiserat. På samma sätt blir nu Javaapplets standarden för exekverbart innehåll: en del av en websida är ett exekverande Javaprogram som tillför intelligens på klientsidan och skapar levande websidor. Med exekverbart innehåll skapas helt nya möjligheter för vad som kan åstadkommas via Internet:

- Animering: en Javaapplet kan skapa animeringseffekter.
- Interaktivitet: en Javaapplet kan visa gränssnittskontroller som knappar, listboxar, editeringsfält etc. som användaren kan integrera med och styra vad som skall hända. Logiken för att hantera detta interagerande kan i sin helhet skötas på klienten, utan att servern behöver vara inblandad och belastas.
- Äkta klient-server system: genom en Java applet kan klient server system enkelt utvecklas där användaren via webläsaren laddar ner sin klientapplikation. Internet blir en global distributionskanal för klient-serversystem där vilken dator som helst är potentiell klient i ett visst system, utan krav på specifik installation på klienten.
- Dynamiska utökningsbara system: en applet kan inte bara ladda ned filer med text, grafik och ljud, den kan också ladda ned andra Javaklasser och exekvera dessa. Detta innebär att en applet dynamiskt kan välja vilka andra Javaklasser som skall användas för att utöka programmets funktionalitet och denna teknik möjliggör också enkel uppdatering av nya versioner. Det finns så kallade innehålls- och protokollhanterare där en webläsare dynamiskt kan ladda ner Javaklasser från en

server, där dessa klasser sedan kan tolka en fil med ett nytt innehåll eller ett kommunikationsprotokoll som webläsaren inte på förhand förstår.
- I och med att Java är uppdelat i olika klasser/objekt går det att ändra i de enskilda klasserna utan att påverka hela applikationen.

2.3.1 Historik

Java offentliggjordes och lanserades officiellt i november 1995, då även Netscape meddelade att man licensierat språket och avsåg att ge fullt stöd för Java i sin webläsare. Ursprungligen fanns endast kärnan i form av själva språket tillsammans med de erfarenheter som Java-projektet fått genom utvecklingen av en webläsare (HotJava) skriven i Java. Efter detta har API:erna och produkterna kring Java växt, vilket har gjort att många alltmer ser Java som en plattform och inte endast ett språk.

Den första versionen av Java använde en kommandoradsbaserad kompilator från Sun där allting kompilerades i ett kommandofönster. Källkoden skrevs in med en vanlig texteditor. Numera finns dock avancerade och integrerade utvecklingsmiljöer som innehåller både kompilator, avlusare och GUI-byggare.

Varifrån kommer namnet Java? Utan att någon avslöjat detaljerna är det sannolikt att namnet föddes vid morgonkaffet…

2.3.2 Tekniska plattformar

Java kan exekveras på i princip alla operativsystem och typer av hårdvara. Allt som behövs är att det finns en virtuell maskin. Exempel på operativsystem: Windows XX, Solaris, MacOs, OS/2, Netware och Linux.
Microsoft har i USA blivit åtalade för att bl.a. ha förvanskat sin version av Java för Windows. De har infört nya egenskaper som Sun (äger rättigheterna till Java) inte har godkänt. I och med dessa nya egenskaper är Microsofts version inte plattformsoberoende.

2.3.3 Underhåll

Utvecklingsmiljöerna för Java innehåller ofta stöd för interaktiv, visuell programmering där delar av programmet kan utvecklas genom att användaren drar och släpper komponenter på fönster och visuellt förbinder objekt med varandra. Stora delar av programmet kan alltså utvecklas genom att användaren visuellt ritat upp sitt användargränssnitt istället för att skapa det med kod direkt. Denna teknik har tidigare

funnits för C++, men det har visat sig att tekniken kan användas i betydligt högre utsträckning för Java. Anledningarna till detta är flera:

- Java är standardiserat: det är enklare att automatgenerera generell kod i Java. Det är entydigt bestämt vilket beteende varje kodrad i Java kommer att skapa.
- Användargränssnittet beskrivs i kod: i Java används inte den komplicerade tekniken med separata sk. resursfiler som exempelvis Windows använder. Ett fönster kan enkelt och entydigt beskrivas med hjälp av genererad Javakod. Om Javakoden sedan modifieras manuellt kan en avancerad miljö återläsa dessa manuella förändringar och uppdatera den visuella representationen av fönstret.
- Java har inkrementell kompilering: bytekoden för varje klass ligger i en egen fil i Java. Om endast ett fönster har uppdaterats behöver endast bytekoden för denna klass uppdateras, och ingen tung omkompilering och länkning behövs. Detta gör att arbetscykeln från förändring till exekvering av programmet är kort.
- Java är plattformsoberoende: applikationen kan utvecklas på en plattform och sedan exekveras på en annan.
- Dynamisk hantering av komponenter: nya komponenter kan laddas in dynamiskt av en utvecklingsmiljö och sedan göras tillgänglig för applikationsutveckling.

2.3.4 Användarvänlighet

Utvecklingsmiljöerna för Java gör att det inte är allt för svårt att arbeta med språket. Det ställer dock krav på användaren att denne skall förstår objektorientering och kunna programmering. Det finns gått om alla sorters litteratur, både i elektronisk och tryckt form.

2.4 Common Gateway Interface (CGI)

(Frykholm, 1997), (Ladd & O'Donnel, 1998), (*CGI – Common Gateway Interface*)

Vanliga HTML-dokument är statiska. Först skrivs dokumentet och sedan läggs det upp på en server, sedan kan alla läsa dokumentet i en webläsare. Om informationen blir inaktuell måste dokumentet öppnas i ett redigeringsverktyg sedan får man, göra de ändringar som krävs och sedan lägga upp det på servern igen. Detta är ofta opraktiskt och ibland omöjligt.

Ett annat problem med vanliga HTML-dokument är att de inte är interaktiva. Allt som skall visas för användare måste kodas in i förväg och kan därför inte påverkas av användaren. Dessa problem löses genom att skicka användarens val till ett program som genererar innehållet i HTML-dokumentet dynamiskt utifrån de önskemål som användare angett. För att det skall fungera måste det finnas en överenskommelse om hur programmen som genererar sidorna skall samarbeta med webservern, så att

programmen kan motta information från användaren och producera websidor åt denne. Det är denna överenskommelse som kallas för CGI. CGI är alltså inte ett program eller ett programmeringsspråk, utan det är en överenskommelse om hur en server och ett program som genererar websidor skall kommunicera.

CGI-program kallas för skript (scripts) eftersom de första CGI-programmen skrevs med UNIX shell scripts och Perl. När ett Perl-program startas översätts instruktionerna till maskinkod i just det ögonblicket och är alltså inte kompilerat. I det avseendet är Perl ett skript som översättaren följer. Andra språk, som C, är kompilerade i förväg och kallas därför inte skript. I CGI-sammanhang kallas däremot både kompilerad och okompilerad kod för skript.

När en användare via sin klient (webläsare) är intresserad av ett visst dokument med CGI skript så kopplar klienten upp sig mot webservern och begär sidan. Servern undersöker den begärda adressen för att se om den leder till ett CGI-program. För att utröna det använder den sig av vissa regler som specificeras i inställningsfilerna. Om servern avgör att dokumentet är ett CGI-program så körs skriptet och resultatet av körningen skickas till klienten som svar på dess förfrågan. (Om servern inte godkänner dokumentet som ett program skickas istället programkoden som om det var en vanlig HTML-fil.)

Innan servern startar skriptet skapar den ett antal systemvaribler, för serverns status, som lämnas till skriptet och systemvariablerna försvinner när skriptet är klart. Varje skript får en egen unik uppsättning variabler. I praktiken har en aktiv server många skript igång samtidigt, var och en med sin egen miljö. Skriptet läser systemvariablerna och STDIN (standard input) på det sätt den skall. Sedan utför den de moment den innehåller och lämnar sina utdata till STDOUT (standard output). MIME-koderna (Multimedia Internet Message Extentions), som servern sänder till webläsaren, talar om för denna vilken typ av fil som är på väg över nätet. Eftersom denna information föregår själva filen kallas den header. Servern kan inte skicka någon header för den information som skapas i skriptet, eftersom skriptet kan innehålla text, bilder, ljud och animeringar. Därför är skriptet självt ansvarigt för att skicka en header. Alltså måste skriptet utöver att sända sin egen utdata även skicka header-informationen. Om webläsaren inte får eller får en felaktig header kan den inte använda skriptets utdata.

En grov förenkling av hur det fungerar:
Webläsaren tolkar en URL och kontaktar servern.
Webläsaren begär ett dokument från servern.
Servern översätter URL-innehållet till sökväg och filnamn.
Servern förstår att det rör sig om ett program och inte en fil.
Serven förbereder miljön och startar skriptet.
Skriptet startar och läser in systemvariabler och STDIN.
Skriptet sänder rätt MIME-header för innehållet till STDOUT.

Skriptet sänder resten av sin utdata till STDOUT och avslutas.
Servern märker att skriptet är färdigt och stänger förbindelsen.
Webläsaren visar dokumentet.

2.4.1 Historik

Den nuvarande versionen av CGI standarden är version 1.1 och den kom i mitten av 1990-talet.

2.4.2 Tekniska plattformar

CGI skript kan skrivas i vilket språk som helst och finnas på alla plattformar. De vanligaste språken är Perl och C/C++, men även Python, Visual Basic och AppleScript används.

2.4.3 Underhåll

Då CGI skrivs med funktionella språk som inte använder objekt är det svårare och mer tidsödande att göra förändringar i och med att dessa kan påverka större delar i applikationen.

2.4.4 Användarvänlighet

Det finns omfattande litteratur om CGI, från medelanvändare till avancerade.

2.5 Macromedia Flash

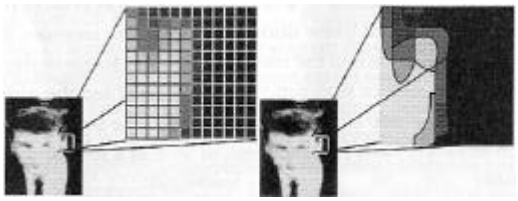
(Plant, 1998), (*Macromedia Flash*), (Lentz & Reinhardt, 2000)

Flash är ett program med vars hjälp det blir enkelt att utnyttja snabb vektorgrafik för att skapa ljud och bildfilmer för webläsare. Programmet innehåller tre delar: Ett vektorbaserat ritprogram, Ett animeringsverktyg för att skapa eller simulera rörelser på skärmen och ett verktyg för att skapa interaktiva multimediafiler (filmer).

Vektorbaserat ritprogram

Fördelen med att rita alla figurer i vektorformat är dels att dessa nästan alltid tar upp mindre lagringsutrymme än en bild i t.ex. bitmap eller jpg format och dels att en

vektorbild kan förändras i storlek utan att förlora i upplösning/form, vilket den gör i de andra bildformaten.



Figur 2:1. Bilden till vänster är i bitmapformat och när den förstoras bildas ett rutnät. Bilden till höger är en vektorbild som behåller sitt utseende när den förstoras (Plant, 1998, s 4).

Anledningen till denna olikhet i upplösning/form beror på hur bilden är uppbyggd. Vektorprogram använder matematiska formler för att beskriva former och linjer hos en figur och detta gör också att bildens storlek kan förändras utan att filstorleken ökar speciellt mycket. De andra bildformaten beskrivs med hjälp av prickar (pixels) i ett rutnät. Varje pricks färg lagras i minnet antingen som ett nummer på en fördefinierad lista eller som mängden rött, grönt och blått den innehåller. Detta gör att bildens filstorlek är direkt proportionell mot bildens storlek.

Eftersom bilder på en skärm består av pixels i rutnät, blir alltid horisontella och vertikala linjer bra, men diagonala linjer blir ofta hackiga. Detta kallas för aliasing och följden blir att tunna diagonala linjer och fält i olika nyanser t.o.m. kan försvinna helt, vilket är ett stort problem vid bildhantering.

I Flash finns det en funktion som kallas för ”anti-aliasing” som jämnar ut kanterna mellan pixels med olika färger genom att räkna ut genomsnittsfärgen mellan dem för att skapa en effekt som lurar ögat att uppfatta konturen eller färgövergången som mjuk.

En annan bra funktion i Flash är ”Trace bitmap” som omvandlar en bitmap bild till en vektorbild.

Det är dock inte i första hand ritfunktionerna i Flash som har gjort programmet till en standard som stöds av de flesta webläsare, utan det är animerings- och multimediafunktionerna.

Att animera är att skapa en rörelse genom att flytta ett objekt eller byta ut det över tiden och det är i funktioner för detta som Flashs styrka sitter. En Flashfilm består av ett antal ramar (frames) och det finns tre typer av frames, key-, blank- och tweened frames. Filmen spelas genom att en frame i taget visas. En keyframe innehåller en förändring av ljud eller bild medan en blankframe inte innehåller någon ny information och därför visas föregående frame en gång. Om funktionen tweening används skapar Flash tweened frames som är beräkningar av förändringar mellan två keyframes.

Denna funktion gör att användaren inte själv behöver rita alla mellanlägen. Detta i sin tur har två stora fördelar gentemot andra animeringsprogram. Det är ett snabbt sätt att animera och animationen tar inte upp så mycket minnesutrymme, då den innehåller beräkningar av mellanbilderna i stället för bilderna självt. I Flash byggs varje bild upp av olika lager och detta gör det givetvis lättare att skapa en snygg design. Den stora fördelen med denna teknik är filmen blir mindre då endast en förändring i ett visst lager tar upp minnesutrymme men t ex en bakgrundsbild inte behöver sparas flera gånger.

Med Flash går det också att inkludera ljud i filmen på ett enkelt sätt och den typ av ljud som då används är samples.

En fördel med Flash är att det går att skapa händelser och kommunicera med webläsaren på användarens initiativ utan att behöva skriva någon HTML-kod. Det går t ex att lägga in en knapp i filmen som, när den trycks ner, begär en URL eller startar en ljudslinga.

En annan fördel med Flash gentemot andra typer av animeringar är att Flash använder sig av något sk. streaming. Detta innebär att filmen kan börja visas trots att den inte är helt nerladdad till webläsaren och laddningen fortsätter under tiden.

2.5.1 Historik

De två senaste versionerna av Flash är 3 och 4 som kom i april 1998 respektive juni 1999.

2.5.2 Tekniska plattformar

Flash finns idag (*Macromedia Flash*) installerat i 85,6 % av världens webläsare och finns till alla plattformar. Flashspelaren finns även att ladda ner kostnadsfritt till följande operativsystem: Mac, Windows, Linux, Solaris och Java. Programmets storlek är endast 100 Kb.

2.5.3 Underhåll

För att skapa och underhålla Flash krävs att Macromedias Flash-utvecklingsprogram är installerat på datorn.

2.5.4 Användarvänlighet

Utvecklingsmiljön är grafisk och därför relativt lättarbetad.
Programmeringsmöjligheterna är begränsade. Det finns väl utformade och interaktiva manualer inkluderade i utvecklingsprogrammet.

2.6 XML

(*Extensible Markup Language (XML)*), (North, 1999)

XML står för Extensible Markup Language. XML härstammar från SGML (se även 2.1). SGML togs ursprungligen fram för att tillgodose de grundläggande behoven av att kunna göra datalagring oberoende av något speciellt programpaket eller någon specifik programleverantör. SGML är ett metaspråk, dvs. ett uppmärkt språk. HTML är ett sådant uppmärkt språk och kan därför kallas för SGML-applikation. SGML har deklarationer (det har även XML) som specifierar vilka tecken som skall tolkas som data och vilka som skall tolkas som markeringar. Med hjälp av SGML:s deklarationsregler och resultatet av en informationsanalys kan utvecklaren av SGML-applikationer identifiera olika typer av dokument som till exempel rapporter, broschyrer, tekniska manualer och så vidare, för att utarbeta en Document Type Definition (DTD) för varje dokument. DTD är en funktion i SGML som gör att innehållet kan beskrivas. DTD är kärnan i en SGML-applikation och hela slutresultatets framgång eller misslyckande beror på hur väl DTD:n gjorts. SGML är komplicerat och tidskrävande att utveckla, därför har inte SGML blivit tillräckligt utbrett för att kunna användas som internet/intranet lösning.
XML använder de SGML-funktioner det behöver och försöker införliva erfarenheter dragna från HTML-användningen. Mål som är uppfyllda med XML:

- XML kan användas med existerande webprotokoll (t.ex. HTTP och MIME) och mekanismer, som URL:er och det kräver inga ytterligare tillägg. XML har utvecklats med webben i åtanke, där SGML-funktioner som varit för svåra att använda på denna utelämnats, medan funktioner som behövs på webben antingen har lagts till eller kopierats från applikationer som redan fungerar.
- XML stöder ett brett utbud av applikationer. Det är svårt att stödja ett stort antal applikationer med bara HTML på grund av tillväxten av skriptspråk (JavaScript etc.) som inte följer en allmän standard och då HTML är oflexibelt. XML har fått SGML:s allmänna kärna, men har fått en ökad flexibilitet som gör det verkligt tänjbart.
- XML är kompatibelt med SGML och de flesta SGML-applikationerna kan konverteras till XML.
- Det är lätt att skriva program som bearbetar XML-dokument. En av HTML:s stora fördelar är att det är så lätt även för ickeprogrammerare att sätta samman några rader skriptkod som möjliggör enklare bearbetning. HTML innehåller även några egna funktioner som gör det möjligt att utföra enklare bearbetning som t.ex.

formulär och CGI-frågesträngar. Utvecklarna av XML har tagit intryck av HTML:s framgångar och har försökt bibehålla HTML:s enkelhet genom att utelämna en hel del av SGML:s mer komplexa funktioner.

- Antalet tillvalsfunktioner i XML har hållits på ett absolut minimum. SGML har många tillvalsfunktioner vilket innebär att SGML-program måste stödja dem alla. Den här graden av funktionsrikedom påverkar också komplexiteten vilket även betyder ökad storlek, ökade kostnader och innebär långsam bearbetning.
- XML-dokument kan vara någorlunda självförklarande, utvecklaren kan t.ex. själv namnge taggar. Detta gör att det går att skriva XML utan någon editor. Det går skriva ut vilket XML-dokument som helst och tolka innehållet och dess betydelse, men det går längre än så. Ett korrekt XML-dokument innehåller bl.a. följande:

    1. Beskriver de strukturella reglerna som uppmärkningarna följer.
    2. Listar alla externa resurser (externa enheter) som dokumentet består av.
    3. Deklarerar alla interna resurser (interna enheter) som används i dokumentet.
    4. Listar icke-XML-resurser (notationer) som används och identifierar alla hjälpapplikationer som kan krävas.
    5. Listar alla icke-XML-resurser (binära) som används i dokumentet och identifierar alla hjälpapplikationer som kan krävas.

- XML:s design är formell och koncis. Grunden i XML-specifikationen utgörs av Extended Backus-Naur Format (EBNF) vilket är en metod som en majoritet av programmerare anser vara enkel att förstå. Information som markerats i XML kan lätt bearbetas av andra program. Dessutom är det relativt enkelt för programmerare att utveckla program som fungerar tillsammans med XML tack vare EBNF-systemet som är känt för programmerare och som dessutom är i det närmaste otvetydigt.
- XML-dokument är lätta att skapa. HTML är välkänt för att vara lättavändbart och XML-utvecklarna drog nytta av detta faktum genom att låta XML likna HTML i sin uppbyggnad med taggar. Det är till och med lättare att skapa ett XML-dokument än ett HTML-dokument, eftersom dokumentskaparen inte behöver lära sig några markeringstaggar utan kan skapa sina egna.

XML tillför även ett antal funktioner som gör språket mycket mer lämpat för internet än både SGML och HTML:

- Modularitet: Trots att HTML inte förefaller ha någon DTD finns en indirekt DTD inlagd i webläsaren. SGML har ingen begränsning i antalet DTD:er, men det finns trots allt bara en för varje typ av dokument. I XML är det möjligt att välja att avstå från att använda DTD helt och hållet eller att använda dem för att skapa sofistikerade mekanismer. Dessa mekanismer är t.ex. kombinationer av multipla

fragment av antingen XML-instanser eller separata DTD:er i en sammansatt instans.

- Tänjbarhet: XML:s kraftfulla länkmekanism gör det möjligt att länka till material utan att länken behöver vara fysiskt kopplad till objektet. Detta öppnar spännande möjligheter där man kan länka ihop saker som t.ex. enheter och material som man inte har skrivrättigheter till. Exempel på detta är CD-ROM-skivor, kataloger, resultatet av en databassökning eller till och med icke-dokumentmaterial som ljudfragment och delar av videofilmer. Dessutom kan man spara länkarna separat skilda från objekten. Detta ger helt andra förutsättningar att underhålla länkar på längre sikt.
- Distribution: Förutom länkhanteringen tillför XML en ännu mer sofistikerad metod att lägga in länkar i den gällande instansen. Detta öppnar dörren till en ny värld av sammansatta dokument, d.v.s. dokument som skapas av fragment av andra dokument som sätts samman automatiskt när de efterfrågas och visar vad som är aktuellt vid just det tillfället. Innehållet kan på detta sätt anpassas efter tidpunkt, media eller till läsaren och existerar därmed bara flyktigt, d.v.s. virtuell information sammansatt av virtuella dokument.
- Internationellt: Både HTML och SGML är beroende av ASCII vilket gör användningen av nationella tecken svår. XML baseras på Unicode och kräver att all XML-mjukvara stödjer Unicode. Unicode ger inte bara XML möjlighet att hantera de tecken som ingår i de västerländska språken, utan klarar även av asiatiska språk.
- Dataorienterat: XML är i första hand dataorienterat även om det också är läsbart för människor. Att XML skulle vara läsligt för människor var ett av XML:s designmål.

2.6.1 Historik

Arbetet med XML startade i juni 1996 och i februari 1998 kom W3C:s XML 1.0 rekommendation. I januari 1999 kom rekommendationen för namnutrymmen (som är en metod för att beteckna de namn som används i ett XML-dokument genom att associera dem med innehållet som identifieras av systemidentifierare) i XML. I juni 1999 kom rekommendationen om Style Sheet Linking. Rekommendationen för XSL Transformations (XSLT, ett språk för att flytta XSL-specifikationer från ett XML-dokument till ett annat. XSL är ett style sheet språk för XML) och XML Path language (Xpath, ett språk för att adressera till delar av XML-dokument) kom i december 1999.

2.6.2 Tekniska plattformar

Förra året, 1999, fanns det i princip utvecklingsverktyg för XML i alla miljöer, men bara Internet Explorer 5.0 stöder XML 1.0 rekommendationerna helt. Redan Internet Explorer 4 hade visst stöd för XML. Från och med nästa version (6) av Netscape Navigator kommer det att finnas fullt stöd för XML. Även Opera 4.0 (norsk webläsare) kommer att stödja XML.

2.6.3 Underhåll

XML-dokument är enkla att underhålla och uppdatera av den anledningen att de är välstrukturerade.

2.6.4 Användarvänlighet

XML är ett språk som kan användas för att göra avancerade applikationer, men även för den som inte är programmerare går det att utveckla enklare applikationer. En fördel med XML ur inlärningssynpunkt är att nybörjare inte behöver kunna en massa taggar, eftersom man kan skriva dem själv. Antalet böcker om XML växer stadigt och utvecklingsverktygen blir fler och fler.

2.7 Java Server Pages

(*Java Server Pages™ Technology*)

Java Server Pages (JSP) är liksom Java skapat av Sun Microsystems och ett antal samarbetspartners. I gruppen samarbetspartners ingår tillverkare av webservrar, applikationsservrar, transaktionssystem och utvecklingsverktyg. Sun utvecklade JSP specifikationen för att kunna integrera och stödja andra Java tekniker såsom Java Servlets (program på servern) och Java Beans (komponenter).

JSP ökar utvecklingstakten av dynamiska websidor på ett antal olika sätt:

- JSP-teknologin delar upp innehållsgenereringen från weblayoten: Med JSP-teknologi använder utvecklaren HTML- eller XML-taggar för att designa och formatera de sidor som skall visas i webläsaren. Utvecklaren använder JSP-taggar eller scriptlets (korta skript) för att generera dynamiskt innehåll på sidan (innehåll som förändras beroende på vilket innehåll som begärts av användaren). Logiken som genererar innehållet är inkapslad i taggar och Java Beans komponenter och sammanbundna till scriptlets som körs på servern. Andra personer än utvecklaren av källogiken kan editera och arbeta med JSP-sidorna utan att det kan påverka

genereringen av innehållet, eftersom källogiken är inkapslad i taggar och Java Beans skilt från layouten.

På servern tolkar en JSP-motor JSP-taggar och scriptlets för att sedan generera det innehåll som begärts och skicka tillbaka resultatet som en HTML- eller XML-sida. Detta hjälper utvecklare att kapsla in kod och samtidigt försäkrar sig om att sidorna är helt portabla.

- JSP bygger på återanvändbara komponenter: De flesta JSP-sidorna använder återanvändbara, plattformsoberoende komponenter för att genomföra de mest komplexa beräkningarna/bearbetningarna som krävs av webapplikationen. Utvecklare kan dela med sig eller byta komponenter med varandra för att utföra gemensamt arbete, eller kanske dela med sig till grupper som är intresserade av att köpa dem. Det komponentbaserade angreppssättet gör att den allmänna utvecklingshastigheten ökar och att utvecklarna får säkrare/stabilare kod.

- JSP förenklar hemsidestillverkning med hjälp av taggar: Hemsidesutvecklare är inte alltid programmerare med kunskap i skriptspråk. JSP-teknologin inkapslar mycket av funktionaliteten som behövs för skapandet av dynamiskt innehåll och gör funktionaliteten tillgänglig via enkla JSP-specifika XML-taggar. Standard-JSP-taggar används till att komma åt och instantiera Java Beanskomponenter, sätta och hämta attribut, ladda ner applets och utföra andra funktioner som i vanliga fall är svårare och mer tidskrävande att programmera.
  JSP-teknologin kan byggas ut genom att skapa skräddarsydda ”taggbibliotek”. Det finns taggbibliotek för vanliga funktioner att köpa, vilket gör att utvecklare inte behöver göra allt själv.

JSP kan stödja komplexa webbaserade applikationer tack vare dess intima integration med Java och dess undergrupper.

2.7.1 Historik

JSP:s första specifikation fick beteckningen 0.91 och den kom i början av 1998. Under sommaren samma år kom version 0.92 . Version 1.0 kom hösten 1998 och version 1.1 kom i november 1999.

2.7.2 Tekniska plattformar

Java Server Pages kan visas i alla webläsare eftersom det är HTML som skickas från servern (även XML kan användas som presentationsspråk, men stödet i dagens webläsare är dåligt). Serverdelen av JSP ingår i ett antal olika webserverprogramvaror,

det finns dessutom en liten webserver (för nerladdning på www.sun.com) som är gjord för JSP, skriven i Java och som är gratis.

2.7.3 Underhåll

Ett av de starkaste argumenten för JSP är att det är enkelt att uppdatera och underhålla tack vare att layout och kod är skilda åt. En nackdel är att det bara finns ett fåtal grafiska editorer som stöder JSP, men antalet ökar troligen snabbt, med tanke på dagens tekniska utveckling.

2.7.4 Användarvänlighet

Det finns än så länge ett begränsat utbud av litteratur eftersom JSP är en ny utvecklingsteknik. Presentationsdelen är för den vane webdesignern inget problem, men den logiska delen kräver att man har programmeringskunskaper.

2.8 Active Server Pages

(Francis, Fedorov, Harrison, Homer, Murphy, Sussman, Smith & Wood, 1998), (*ASP White Papers*), (*Chili!Soft | Chili!Soft ASP*)

ASP betyder Active Server Pages och innebär att all kod körs på servern. Med andra ord är man alltså helt oberoende av vilken webläsare användaren kör, oavsett tillverkare eller version. Allt genereras i serverns programvara. När det sedan presenteras för användaren är det HTML-kod som man kommer att se ifall man undersöker källan.

En ASP-sida är en textfil med filändelsen **.asp** som innehåller HTML, och kan innehålla skript för klienten och servern. När användaren skall besöka en sida som innehåller ASP-kod sker följande:

1. Användaren efterfrågar sidan.
2. Filen bearbetas i serverns programvara som läser ASP-koden och utför vad koden ber om.
3. Koden, som nu, förhoppningsvis genererat någonting, tas bort från filen och allt som presenteras för användaren i webläsaren är helt vanlig HTML.

Koden eller skriptet, som är ett mera korrekt uttryck för detta, kräver särskild programvara på servern för att det skall fungera. Det vanligaste är att servern använder Microsoft Internet Information Server (IIS) som programvara. Microsofts

lättversion av denna programvara, Microsoft Personal Webserver, har begränsade funktioner jämfört med IIS, men den duger alldeles utmärkt för att köra ASP på en vanlig persondator.

ASP körs på servern, men det finns ingen kod som egentligen kallas ASP. All kod som skrivs är VBScript (Visual Basic Script) eller Jscript (JavaScript). Det går att använda båda dessa två språk för att åstadkomma ASP-sidor.VBScript är en bantad version av Microsofts riktiga VisualBasic-språk.

ASP kan användas till bland annat:

- Hämta och lagra information i en databas via websidor
- Göra sidorna mer interaktiva för dina besökare. Ett enkelt exempel på detta är att visa aktuell tid och datum på sidan då användaren surfar in.
- Göra sidorna mer personliga, t.ex. genom att låta användaren välja vilken storlek man skall ha på texten.
- Lagra information om en användare som kan användas varje gång användaren kommer tillbaka för att på så sätt kunna påminna henne om något eller för att vissa personliga inställningar skall synas.
- Göra spel, gästböcker, annonsmarknader, skapa och läsa information från cookies etc.

När det talas om webutveckling brukar ordet webapplikation nämnas. En webapplikation kan t.ex. vara en samling ASP-sidor och serverkomponenter. En website kan innehålla flera av dessa. Varje sådan applikation har sin egen uppsättning variabler och attribut som beskriver dess nuvarande tillstånd. Dessa variabler och attribut förändras under hela applikationens livstid, från början och tills att den sista sessionen är stängd. Problemet med webbaserade applikationer har hela tiden varit att HTTP, som är det underliggande protokollet, har dåligt med minnesresurser och sparar ingen information från den ena till den nästföljande tjänsten en klient (webläsare) begär. Detta har lösts i ASP genom att man här använt sig av objekt som kan hålla information lagrad dels under hela tiden applikationen är igång och dels under en användares session.

För att skapa dynamiska och interaktiva ASP-sidor finns ASP:s objektmodell med fem objekt som hjälp för utvecklaren. Dessa är **Request, Response, Application, Session** och **Server.** Objekten har varsin uppsättning metoder, egenskaper och händelser som innehåller all funktionalitet hos objekten. Request- och Response-objekten används för att kommunicera mellan klienten (webläsaren) och servern. Request används för att ta emot information och Response för att skriva ut. Application- och Session-objekten används för att lagra information i det virtuella minnet hos servern. Server-objektet är till för att påverka inställningar på servern, t.ex. finns det en metod för att skapa instanser av ASP-komponenter.

Scripting Objects är tillgängliga om VBScript väljs som skriptspråk för ASP-sidorna. Till Scripting Objects hör FileSystem-, TextStream- och Dictionaryobjekten.

2.8.1 Historik

I juli 1996 tillkännagav Microsoft att en teknologi för att skriva skript på servern var på väg. Den första betaversionen kom i november 1996 och den följdes en månad senare av den första skarpa versionen, ASP 1.0 som släpptes tillsammans med Internet Information Server (IIS) 3.0 . I december 1997 kom version 2.0 av ASP tillsammans med IIS 4.0. Version 3.0 av ASP kom när IIS 5.0 (som kom med Windows 2000) släpptes i februari 2000.

2.8.2 Tekniska plattformar

Från början kunde bara ASP köras på Windows NT, men företaget Chili!Soft har en parser som gör att ASP kan köras på andra webservrar än IIS. Deras version går under namnet Chili!ASP. ASP i sig självt är plattformsoberoende för klienterna, men om nyare versioner av HTML används kan det uppstå en del problem med äldre webläsare.

2.8.3 Underhåll

ASP-kod är inte svår att underhålla och uppdatera eftersom den använder sig av skriptspråk. Det finns bra stöd för ASP i utvecklingsverktyg för internetrelaterade programmeringsspråk. De mer avancerade lösningarna med ASP kräver programmeringskunskap.

2.8.4 Användarvänlighet

Det finns mycket böcker både för nybörjare och professionella användare. Eftersom ASP skrivs med skript är det inte alltför svårt att lära sig skriva ASP-sidor.

3 Analys och jämförelse av olika VT

För att kunna göra en bedömning av vilket VT som är mest lämpligt att använda ur teknisk synpunkt för att producera en interaktiv website, har vi valt att bedöma VT med hjälp av ett antal parametrar. De uppställda parametrarna har vi i vissa fall även delat upp i underparametrar:

- Plattformar (operativsystem och webläsare)
- Kostnad
- Grafiska möjligheter (interaktivitet och animering)
- Användarvänlighet (svårighetsgrad, utvecklingstid och litteratur)
- Resursbelastning (serverbelastning och uppstartshastighet)
- Underhåll

Eftersom utvecklingen av olika VT går väldigt fort visar vi i nedanstående tabell vilken version av respektive VT vi avser.

Verktyg/teknik	**Versionsnummer**
HTML	4.0 (DHTML)
JavaScript	1.3
Java	2
CGI	1.1
Macromedia Flash	4
XML	1.1
Java Server Pages	1.1
Active Server Pages	3.0

Figur 3:1 visar senaste versionsnummer för respektive VT.

3.1 Plattformar

Plattformsparametern har vi delat upp i två delar: operativsystem och webläsare.

3.1.1 Operativsystem

Operativsystemet är intressant ur utvecklingssynpunkt då inte alla VT finns till alla operativsystem. Vi har valt att undersöka de operativsystem som är vanliga på arbetsstationer. Windows är för närvarande det dominerade operativsystemet för arbetsstationer i Sverige, men Mac används oftast i grafiska branscher och Unix är fortfarande vanligt på stora industriföretag.

Verktyg/teknik	Operativsystem	Gradering
HTML	Windows, Unix, Mac	3
JavaScript	Windows, Unix, Mac	3
Java	Windows, Unix, Mac	3
CGI	Windows, Unix, Mac	3
Macromedia Flash	Windows, Mac	2
XML	Windows, Unix, Mac	3
Java Server Pages	Windows, Unix	2
Active Server Pages	Windows, Linux (Unix)	2

Figur 3:2 visar vilka operativsystem som respektive VT stöds av.

Enligt figur 3:2 går de flesta VT att utveckla under de flesta operativsystem, med tre undantag. Flash går inte att utveckla med Unix och varken JSP eller ASP går att köra på Macintoshservrar.

3.1.2 Webläsare

I detta avsnitt undersöker vi på vilka webläsare den färdiga produkten av respektive VT fungerar. Denna produkt kallar vi i fortsättningen för verktygsprodukt (VTP). Vi utgår från den senaste versionen av respektive VT.

Internet Explorer är den vanligaste webläsaren under Windowsmiljö och version 4 och 5 är de som används mest. Netscape är den vanligaste webläsaren under Unix och Mac och här används version 4.x mest. Det är dock viktigt att ha i åtanke att inte alla använder de senaste versionerna (t.ex. levereras Windows NT 4 med Internet Explorer 2).

Verktyg/teknik	Webläsare	Gradering
HTML	IE 4.0 uppåt,Netscape 4.0 uppåt	2
JavaScript	IE 4.0 uppåt,Netscape 4.0 uppåt	2
Java	IE 4.0 uppåt,Netscape 4.0 uppåt	2
CGI	Alla som stöder formulär i HTML	3
Macromedia Flash	IE 3.0 uppåt, Netscape 2.0 uppåt	3
XML	IE 5.0 (bara delvis), Netscape 6 (beta)	1
Java Server Pages	Alla som stöder formulär i HTML	3
Active Server Pages	Alla som stöder formulär i HTML	3

Figur 3:3 visar vilka webläsare som stöder respektive VT.

De VTP som stöds av flest webläsare är CGI, Flash , JSP och ASP. Det är dock viktigt att lägga märke till att Flash VTP kräver plugin i webläsaren. Denna plugin (senaste versionen) är standard from IE 4 och Netscape 4 utom i Unix.

3.2 Kostnad för utvecklingsverktyg

I vår tabell figur 4:4 visar vi den lägsta kostnaden för respektive VT, men det finns avancerade verktyg för utveckling i olika VT som är mycket dyrare.

Verktyg/teknik	Kostnad	Gradering
HTML	Gratis	3
JavaScript	Gratis	3
Java	Gratis för enkla verktyg	2
CGI	Gratis för enkla verktyg	2
Macromedia Flash	ca 4000 SEK/licens	1
XML	Gratis för enkla verktyg	2
Java Server Pages	Gratis för enkla verktyg	2
Active Server Pages	Gratis för enkla verktyg	2

Figur 3:4 visar den lägsta kostnaden för respektive VT.

Det enda verktyg som alltid kostar pengar är Macromedia Flash men om man skall bedriva utveckling med hjälp av de andra VT är det lämpligt att köpa ett mer avancerat utvecklingsverktyg. Detta är inte så viktigt när det gäller utveckling av HTML då det är ett relativt enkelt VT.

3.3 Grafiska möjligheter

Grafiska möjligheter delar vi upp i två delar: interaktivitet och möjlighet till animering.

3.3.1 Interaktivitet

Interaktiviteten är en mycket viktigt del i t.ex. websiter som innehåller manualer, då lärandet i hög grad påverkas av hur delaktig användaren kan vara (Allwood, 1991). Beroende på användarens kompetens på produkten som manualen beskriver vill denne utnyttja manualen på olika sätt.

Vi har graderat möjligheten till interaktivitet hos VTP:n på en skala 1 till 3:

1. I första hand en statisk presentation av information och bild, där användaren endast kan välja vilken presentation man vill se.

2. VTP kan ge informationsrespons på användarens aktiviteter, men inte någon direkt grafisk respons.

3. VTP kan styras helt beronde på användarens handlingar.

Verktyg/teknik	**Interaktivitet**
HTML	1
JavaScript	3
Java	3
CGI	2
Macromedia Flash	3
XML	2
Java Server Pages	2
Active Server Pages	2

Figur 3:5 visar möjligheterna att skapa interaktivitet med hjälp av respektive VT.

Som synes i figur 3:5 är JavaScript, Java och Flash de VT som gör det möjligt att skapa den bästa interaktiviteten, men JavaScripts 3:a är svag. Detta beror på att det inte går att påverka en bild direkt i samma bild.

3.3.2 Möjlighet till animering

Möjligheten till att skapa animeringar är viktig när det skall beskrivas ett händelseförlopp för användaren. Om användaren skall utföra ett komplext händelsemönster för att uppnå ett visst mål är det lättare för användaren att lära sig detta om man får se hur det går till genom en animering, än att läsa om det i en text (Allwood, 1991).

Verktyg/teknik	Möjlighet till animering
HTML	1
JavaScript	2
Java	3
CGI	1
Macromedia Flash	3
XML	1
Java Server Pages	1
Active Server Pages	1

Figur 3:6 visar möjligheten att skapa animering med hjälp av respektive VT.

JavaScript kan användas till att växla bilder automatiskt. Java och Flash är de VT som kan användas till att göra de bästa animeringarna av de VT vi valt att studera. Med övriga VT måste användaren själv se till att animeringen sker genom att klicka sig fram mellan bilder.

3.4 Användarvänlighet

Vi har delat upp parametern användarvänlighet i tre delar: svårighetsgrad, utvecklingstid och litteratur.

3.4.1 Svårighetsgrad

Med svårighetsgrad avser vi hur svårt det är att lära sig att behärska respektive VT. Detta är viktigt bl.a. för nya användare då de skall välja VT. Vidare är det viktigt för en organisation att känna till svårighetsgraden, dels då inlärningstiden kostar pengar och dels då personal byts ut och system skall underhållas.

Om ett system skall uppdateras och ingen i organisationen i det ögonblicket behärskar VT:n, dröjer det olika lång tid tills den önskade/nödvändiga uppdateringen kan genomföras.

Svårighetsgraden graderas från 1 till 3, där 3 är enklast att lära sig:

1. Omfattande programmeringskunskaper krävs.

2. Det finns delvis grafiska hjälpmedel och viss programmeringskunskap krävs.

3. Det finns helt grafiska hjälpmedel för utveckling och inga programmeringskunskaper krävs.

Verktyg/teknik	Svårighetsgrad
HTML	3
JavaScript	2
Java	1
CGI	2
Macromedia Flash	3
XML	2
Java Server Pages	2
Active Server Pages	2

Figur 3:7 visar hur enkelt respektive VT är att lära sig.

HTML och Flash är de VT som är lättast att lära sig, HTML då det inte är speciellt omfattande och Flash då allt utvecklas i en enhetlig miljö och nästan uteslutande med hjälp av "drag 'n drop" teknik.

3.4.2 Utvecklingstid

Med utvecklingstid avser vi hur lång tid det tar att producera en lite mer omfattande grafisk presentation som t.ex. en manual. (Utvecklingstid innefattar inte utbildningstid)

Utvecklingstiden graderas från 1 till 3, där 3 är snabbast.

Verktyg/teknik	Utvecklingstid
HTML	1
JavaScript	3
Java	2
CGI	2
Macromedia Flash	1
XML	2
Java Server Pages	2
Active Server Pages	2

Figur 3:8 visar hur lång tid det tar att utveckla i respektive VT, i relativa termer.

I HTML och Flash måste varje sida i websiten skapas unikt av utvecklaren och därför tar dessa längst tid. I de VT som har fått en 2:a kan varje sida genereras av koden (applikationen) men själva applikationen tar relativt lång tid att utveckla. Med JavaScript kan varje del av en grafisk presentation genereras av koden (applikationen) och även själva applikationen går fort att utveckla.

3.4.3 Litteratur

Vi har valt att ta med den tillgängliga mängden litteratur, som ett av våra bedömningskriterier då detta är en viktig aspekt när möjligheten att lära sig nya VT skall bedömas. Finns det inte relevant utbildningsmaterial ökar inlärningsperioden och inlärningströskeln förlängs.

Litteratur graderas från 1 till 3, där 3 är störst tillgänglig mängd:

1. Begränsat med litteratur och nästan bara för medel till avancerade användare

2. Begränsat med litteratur, men omfattande exempelsamlingar på www för alla kunskapsnivåer

3. Omfattande eller obegränsat, för alla kunskapsnivåer

Verktyg/teknik	**Litteratur**
HTML	3
JavaScript	3
Java	3
CGI	3
Macromedia Flash	2
XML	1
Java Server Pages	1
Active Server Pages	3

Figur 3:9 visar litteraturomfattningen för respektive VT.

För alla VT finns exempel på www och omfattningen av dessa beror till stor del på hur länge respektive VT har använts. Det är oftast svårare att hitta det som söks i exempelsamlingar, då katalogiseringen inte är standardiserad. Kvalitén på information som finns på www är mycket varierande då vem som helst kan publicera den utan granskning, medan litteratur i tryckt form åtminstone granskas av förlaget.

Flash är den VT som har minst litteratur och anledningen till detta är att utvecklingsmiljön inte är någon standard utan ägs av Macromedia.

3.5 Resursbelastning

Resursbelastning har vi delat upp i två delar: serverbelastning och uppstartshastighet.

3.5.1 Serverbelastning

Med serverbelastning avser vi den belastning som servern, vilken VTP:n ligger på, utsätts för när flera personer använder VTP:n samtidigt.

Serverbelastning graderas från 1 till 3, där 3 är minst belastning.

Verktyg/teknik	**Serverbelastning**
HTML	3
JavaScript	3
Java	2
CGI	1
Macromedia Flash	3
XML	3
Java Server Pages	2
Active Server Pages	2

Figur 3:10 visar serverbelastning för respektive VTP.

HTML, JavaScript, Flash och XML belastar servern minst tack vare att all kod tolkas i webläsaren och endast de bilder som hör till VTP:n förbrukar resurser vid nedladdning från servern. CGI, JSP och ASP körs hela tiden på servern och belastar således denna en del. CGI använder sig av en mer resurskrävande teknik än JSP och ASP.

3.5.2 Uppstartshastighet

Med uppstartshastighet avser vi den hur lång tid det tar att starta VTP:n för den enskilde användaren.

Uppstartshastighet graderas från 1 till 3, där 3 är snabbast uppstart.

Verktyg/teknik	**Uppstartshastighet**
HTML	3
JavaScript	3
Java	1
CGI	1
Macromedia Flash	2
XML	3
Java Server Pages	2
Active Server Pages	2

Figur 3:11 visar uppstartshastighet för respektive VTP.

Den snabbaste uppstarten fås också med HTML, JavaScript och XML, då de exekveras direkt i webläsaren. Flash körs däremot via en plugin till webläsaren vilket sänker uppstartshastigheten något.

3.6 Underhåll

Med underhåll avser vi hur enkelt det är att underhålla och uppdatera en VTP och hur tidskrävande det är.

Verktyg/teknik	**Underhåll**
HTML	2
JavaScript	2
Java	1
CGI	1
Macromedia Flash	2
XML	3
Java Server Pages	2
Active Server Pages	1

Figur 3:12 visar hur enkelt det är att underhålla och uppdatera respektive VT.

De VTP som kräver mest resurser för att underhålla är Java, CGI och ASP. Anledningen till att Java och CGI förbrukar lite mer resurser är att de kräver mer programmering och anledningen till att ASP förbrukar lite mer resurser är att ASPdokument tenderar att bli mindre strukturerade.

3.7 Sammanfattning

Resultatet av vår analys och jämförelse presenteras i matrisen i figur 3:13 nedan.

	HTML	**Java-Script**	**Java**	**CGI**	**Flash**	**XML**	**JSP**	**ASP**
Plattformar:								
Operativsystem	3	3	3	3	2	3	2	2
Webläsare	2	2	2	3	3	1	3	3
Kostnad	3	3	2	2	1	2	2	2
Grafiska möjligheter:								
Interaktivitet	1	3	3	2	3	2	2	2
Animering	1	2	3	1	3	1	1	1
Användarvänlighet:								
Svårighetsgrad	3	2	1	2	3	2	2	2
Utvecklingstid	1	3	2	2	1	2	2	2
Litteratur	3	3	3	3	2	1	1	3
Resursbelastning:								
Serverbelastning	3	3	2	1	3	3	2	2
Uppstartshastighet	3	3	1	1	2	3	2	2
Underhåll	2	2	1	1	2	3	2	1

Figur 3:13 visar en sammanställning av graderingen av de olika VT med avseende på de olika parametrarna, där 3 är bäst och 1 är sämst.

4 Tolkning av resultatet

I detta kapitel tolkar vi resultatet av analysen och jämförelsen i föregående kapitel. För att tolka resultatet har vi valt att räkna ut ett medelvärde för respektive parameter.

	HTML	**Java-Script**	**Java**	**CGI**	**Flash**	**XML**	**JSP**	**ASP**	**Medel-värde**
Plattformar:									
Operativsystem	3	3	3	3	2	3	2	2	2,63
Webläsare	2	2	2	3	3	1	3	3	2,38
Kostnad	3	3	2	2	1	2	2	2	2,13
Grafiska möjligheter:									
Interaktivitet	1	3	3	2	3	2	2	2	2,25
Animering	1	2	3	1	3	1	1	1	1,63
Användarvänlighet:									
Svårighetsgrad	3	2	1	2	3	2	2	2	2,13
Utvecklingstid	1	3	2	2	1	2	2	2	1,88
Litteratur	3	3	3	3	2	1	1	3	2,38
Resursbelastning:									
Serverbelastning	3	3	2	1	3	3	2	2	2,38
Uppstartshastighet	3	3	1	1	2	3	2	2	2,13
Underhåll	2	2	1	1	2	3	2	1	1,75

Figur 4:1 visar en sammanställning av graderingen av de olika VT med avseende på de olika parametrarna. Figuren innehåller också medelvärdet av graderingen för respektive parameter.

4.1 Plattformar

4.1.1 Operativsystem

Stora organisationer har ofta flera olika datormiljöer (t.ex. Unix, Windows och Macintosh). Detta påverkar ofta valet av ny teknik. I vår jämförelse av VT låg medelvärdet på 2,63 (se figur 4:1). Flash ligger under medelvärdet på grund av att det inte går att utveckla Flashanimeringar i Unixmiljö. Både JSP och ASP ligger under medelvärdet då det inte finns stöd för dem på Macservrar. Dessa tre avvikelser är dock acceptabla dels då den grafiska delen av websiter vanligtvis inte utvecklas på Unix och dels då Macservrar är relativt sällsynta.

4.1.2 Webläsare

En website på internet, men framförallt på intranet, bör vara tillgänglig för alla användare för att så många som möjligt skall kunna ta del av den. Idag används flera olika versioner av framförallt Netscape och Internet Explorer vilket innebär att VTPn måste kunna tolkas av de flesta webläsarna. Utvecklaren kan dock välja att inte ta hänsyn till de allra äldsta webläsarna då dessa är väldigt sällsynta och hänsyn till dessa skulle kraftigt begränsa websiten. Medelvärdet för denna parameter var 2,38 (se figur 4:1). XML avvek mest av alla VT och har endast delvis stöd i de allra senaste webläsarna. Detta gör att XML idag är ett olämpligt VT. HTML, JavaScript och Java ligger också de under medelvärdet, men denna avvikelse är acceptabel då de endast stöd i de äldsta webläsarna.

4.2 Kostnad

Medelvärdet för denna parameter var 2,13 och de flesta VT var gratis. Det VT som avviker mest var Flash som har ett pris på ca 4000 SEK per licens. Detta är för de flesta utvecklare en låg kostnad, men för väldigt stora organisationer med många utvecklare kan detta bli en aspekt att ta hänsyn till. Denna låga kostnad tycker inte vi är något skäl att välja bort Flash.

4.3 Grafiska möjligheter

4.3.1 Interaktivitet

Interaktivitet är en mycket viktig del i websiter för då användaren är delaktig har denne mycket lättare att ta till sig och komma ihåg information. Medelvärdet på denna parameter var 2,25 och den som avvek mest var HTML som är nästan helt statisk. HTML är därför inte lämpligt när man avser att producera en interaktiv website. CGI, XML, JSP och ASP låg även de under medelvärdet, men möjligheterna till interaktivitet med hjälp av dessa VT är ändå tillräckligt goda.

4.3.2 Animering

Möjligheten att skapa animeringar är viktig när man skall beskriva en komplex situation, dels för att användaren kan se hur det fungerar och dels genom att websiten kan bli mer underhållande och därmed kan bli bättre på att förmedla information. Medelvärdet på denna parameter är 1,63. De VT som har lägst gradering är HTML, CGI, XML, JSP och ASP. Då möjligheten till animering med hjälp av dessa VT är väldigt begränsad bör de inte användas om man skall producera t.ex. manual- eller underhållningssiter.

4.4 Användarvänlighet

4.4.1 Svårighetsgrad

Medelvärdet på denna parameter var 2,13 och Java var det VT som var svårast att lära sig. Man bör därför tänka sig för innan man väljer detta VT då inlärningskostnaderna är höga och det kan vara svårt att ersätta befintlig personal utan en ny inlärningsperiod.

4.4.2 Utvecklingstid

I vår analys och jämförelse av VT undersökte vi utvecklingstiden när man skall producera lite mer omfattande grafiska websiter och genomsnittet var 1,88. De VT som tog längst tid att utveckla med var HTML och Flash eftersom varje sida i websiten måste skapas unikt. Övriga VT kan generera stora delar av varje sida vilket gör att ju större siten är desto mer tid ”tjänar” man på att använda dessa. JavaScript som fick den högsta graderingen är ett VT där även själva applikationen (kodgeneratorn) går fort att utveckla.

4.4.3 Litteratur

Medelvärdet på denna parameter var 2,38. XML och JSP avvek mest och för dessa finns det begränsat med litteratur och nästan bara för medel till avancerade användare. Dessa VT är idag ofta inte lämpliga att välja om man inte är en erfaren programmerare i och med att det finns ytterst begränsat med litteratur för nybörjare. En anledning till att litteraturen är begränsad är att dessa är relativt nya VT och mängden litteratur för olika nivåer kommer sannolikt att öka.

4.5 Resursbelastning

4.5.1 Serverbelastning

Serverbelastning är en viktig parameter särskilt på välbesökta websiter där många kör samma tillämpningar samtidigt. Medelvärdet på denna parameter var 2,38 och det VT som avvek mest var CGI. Detta gör att CGI är ett mindre lämpligt VT om man vet att man utvecklar en website som kommer att ha många besökare samtidigt.

4.5.2 Uppstartshastighet

Användarna idag blir allt mer kräsna när det gäller hastighet på websiter. Om en site tar lång tid att starta ökar risken att användaren söker information på annat håll eller helt enkelt inte tar del av informationen på grund av otålighet. Medelvärdet var på denna parameter 2,13 och Java och CGI avvek mest. Detta blir en viktig parameter då man vet att de flesta användarna utnyttjar en lite långsammare uppkoppling som t.ex. modem.

4.6 Underhåll

Underhåll är ofta en mycket stor eller tom den största delen vid management av en website, därför är detta är betydelsefull parameter och medelvärdet på denna var 1,75. De VT som fick lägst gradering var Java, CGI och ASP, men man bör generellt sett inte välja bort något av dessa VT bara på grund av denna parameter. Hur stor vikt man lägger vid underhållet beror på flera orsaker, t.ex. hur ofta siten skall uppdateras och hur omfattande uppdateringarna är.

5 Metod

För att kunna svara på frågan ”Vilka tekniker och verktyg finns för utveckling av interaktiva internetlösningar?” gjorde vi en omfattande litteraturstudie. Vi har använt oss av traditionell facklitteratur, tidskrifter och internetpublikationer (lästa våren 2000). Därefter gjorde vi en sammanställning av varje VT. Varje sammanställning består av en allmän beskrivning, historik, tekniska plattformar, underhåll och användarvänlighet.

För att kunna svara på frågan ”Vilka verktyg/tekniker (VT) är lämpliga att använda sig av med avseende på nuvarande tekniska nivå i IT-samhället?” gjorde vi en analys och jämförelse av de olika VT vi sammanställt. Denna gjorde vi med hjälp av sex huvudparametrar som vi sin tur i vissa fall delade upp i underparametrar. Huvudparametrarna var: plattformar, kostnad, underhåll, användarvänlighet, resursbelastning och grafiska möjligheter. Dessa parametrar ansåg vi vara viktiga utifrån våra tidigare erfarenheter på området.

En alternativ metod för sammanställning av VT kunde ha varit att intervjua personer i it-branschen. Detta kunde också ha använts i kombination med vår metod.

Alternativa metoder för analysen och jämförelsen av de olika VT kunde ha varit att söka relevanta parametrar i facklitteratur eller att intervjua webutvecklare eller kombinationer av dessa.

6 Slutsatser

Att vi inriktade oss på grafiska websiter med interaktivitet har påverkat analysen av VT:na. Om vi hade inriktat oss mot en mer textbaserad website hade kanske resultatet sett lite annorlunda ut.

XML är inte något lämpligt VT i och med att det saknas stöd i nästan alla webläsare.

HTML bör inte användas som enda VT då man har höga krav på interaktivitet och animering. Utvecklingstiden för HTML är också lång om siten är omfattande.

CGI bör inte heller användas för denna typ av websiter då serverbelastningen är hög och animeringsmöjligheterna är ringa.

JSP och ASP är inga lämpliga VT om möjligheterna till animering är en viktig aspekt för siten, men i övrigt finns det inga speciella hinder för att använda dem.

Java är mycket bra VT för att skapa interaktivitet och animeringar, men ett VT som är lite svårare att lära sig och därmed lite mer krävande att underhålla.

Flashs enda betydelsefulla svaghet är att utvecklingstiden blir relativt lång om websiten är lite mer omfattande. Flash har mycket goda möjligheter till att skapa interaktivitet och animering på ett enkelt sätt.

JavaScript är det enda VT som inte har några direkta svagheter och därför mycket lämpligt att använda vid produktion och management av interaktiva websiter.

Vi anser att resultatet och slutsatserna är trovärdiga tack vare att vi har gjort omfattande litteraturstudier och dessutom valde vi JavaScript när vi producerade en omfattande interaktiv website för EMW.

6.1 Vad kunde ha gjorts annorlunda?

Intervju som redskap för att ta reda på alternativa VT istället för eller i kombination med litteraturstudie hade antagligen givit samma resultat i och med omfattningen av vår litteraturstudie.

I analysen och jämförelsen kunde intervjuer i kombination med litteraturstudier av relevanta parametrar ha givit ett annat resultat då vi kan ha förbisett viktiga bedömningskriterier.

6.2 Fortsatt forskning

Man kan inrikta sig på en speciell parameter som t.ex. underhåll när man jämför de olika VT. Detta då underhåll ofta upptar en stor del av en websites livscykel. Men vilken parameter man än väljer att undersöka mer grundligt går det troligen att få många intressanta resultat på en mer detaljrik nivå, som vi inte kunnat uppnå i vår breda studie. På samma sätt kan man detaljstudera ett speciellt VT.

Ett angränsade område som är intressant är mobila internetlösningar som t.ex. WAP. Med hjälp av denna teknik kan nästan all information i olika websiter bli tillgänglig, inte bara från kontoret eller hemmet, utan i båten, bilen, skogen etc.

7 Källförteckning

Allwood C. M. (1991). *Människa-datorinteraktion; ett psykologiskt perspektiv.* Lund: Studentlitteratur

Allwood C. M. (1998). *Människa-datorinteraktion; ett psykologiskt perspektiv.* Lund: Studentlitteratur

ASP White Papers [WWW dokument]. URL http://support.microsoft.com/support/activeserver/whitepapers.asp

Bishop J. (1997), *Java Gently,* Harlow: Addison Wesley Longman.

CGI - Common Gateway Interface, [WWW dokument]. URL http://www.w3.org/CGI/

Chili!Soft | Chili!Soft ASP, *[WWW dokument]. URL* http://www.chilisoft.com/chiliasp/default.asp

Extensible Markup Language (XML), [WWW dokument]. URL http://www.w3.org/XML/

Fedorov A., Francis B., Harrison R., Homer A., Murphy S., Smith R., Sussman D., & Wood S. (1998), *Professional Active Server Pages 2.0,* Birmingham: Wrox Press Ltd.

Flanagan D. (1997), *Java in a nutshell,* Sebastopol: O'Reilly & Associates Inc.

Frykholm N. (1997), *CGI-programmering,* Upplands-Väsby: Pagina

Goodman D. (1998), *JavaScript Bible,* Foster City: IDG Books Worldwide Inc.

HTML Home Page, [WWW dokument]. URL http://www.w3.org/Markup/

Java Server Pages ™ Technology, [WWW dokument]. URL http://java.sun.com/products/jsp/

Ladd E., & O'Donnell J. (1998), *HTML, Java och CGI,* Hemel Hempstead: Prentice Hall.

Lemay L. (1995), *Teach Yourself Web Publishing with HTML in a week,* Indianapolis: Sams Publishing.

Lentz J., & Reinhardt R. (2000), *Flash 4 Bible,* Foster City: IDG Books Worldwide Inc.

Macromedia Flash, [WWW dokument]. URL http://www.macromedia.com/software/flash/

North S. (1999), *Lär dig XML på 3 veckor,* Upplands-Väsby: Pagina.

Pfaffenberger B., & Gutzman D. (1998), *HTML 4 Bible,* Foster City: IDG Books Worldwide Inc.

Plant D. (1998), *Flash 3! Creative Web Animation,* Berkeley: Macromedia Press

Rice C., & Salisbury I. (1997), *Advanced Java 1.1 Programming,* New York: McGraw-Hill

Solberg H. (1997), *Webdesign,* Jönköping: IDG AB

The Java Tutorial, [WWW dokument]. URL http://java.sun.com/docs/books/tutorial/