TAT Tutorials

Ett läromedel i form av en dynamisk hemsida med tillhörande kod-editor till The Astonishing Tribes programmeringsspråk TAT Cascades.

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
<Institution / avdelning>

Examensarbete:
Jonathan Stahl
Ola Nordström

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds Universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds Universitet
Lund 2006

Sammanfattning

The Astonishing Tribe är ett svenskt teknologi, och design företag som erbjuder produkter och olika sorters service för ökad användarupplevelse vid användandet av mobil telefoni. TATs fokus ligger framförallt i skapandet av användargränssnitt (GUI). Denna rapport är en projektering av ett webbaserat läromedel till TATs XML-baserade programmeringsspråk TAT Cascades.

TAT Cascades är byggt med Kastor som plattform, och tillsammans skapar dessa två programmeringsspråk grafiska användargränssnitt för mobiltelefoner. Dessa gränssnitt är i regel mer dynamiska, snabbare och ger användaren en större multimedia upplevelse än andra gränssnitt. Kort förklarat så sköter Cascades kommunikationen mellan olika komponenter (tex. listor, menyer) och Kastor är dess grafiska renderare. Delar av Cascades består alltså av Kastor, men båda dessa språk är XML-baserade och för en ovan användare är dem svåra att skilja åt.

Under examensarbetet har det skapats en hemsida där handledningstillfällen presenteras. På denna hemsida ska det förklaras hur filstrukturen för olika användargränssnitt är skapad, samband mellan olika komponenter och vad koden betyder. Efter att koden blivit kartlagt går det i medföljande editor, som laddas ner via hemsidan, att se och testköra denna kod. Koden testas genom att trycka på en knapp i editorn som startar upp en simulator. Denna simulator simulerar en mobiltelefons användargränssnitt utifrån koden som blivit skriven i editorn. I slutet av varje handledningstillfälle ska övningar finnas. För att skapa mer interaktivitet och laborerande går dessa övningar att lösa i den medföljande editorn.

Simulatorn som startas via editorn är ett verktyg utvecklat av TAT och var klar redan innan vi började arbeta. Simulatorn kommer inte att diskuteras mer i denna rapport.

Under projekttiden har det producerats en färdigutvecklad prototyp av webbplats, och editor. Denna finns tillgänglig på:
http://www.jonathan-stahl.se/1EXJOBB/login.php

Anhållan om login och lösenord går att skicka till:
ihm04pno@student.lu.se
jonathan.stahl.410@student.lu.se

Nyckelord: Läromedel, interaktivitet, handledning, PHP, SQL, JavaScript, Cascades, Kastor, editor

Abstract

TAT is a Swedish software technology and design company offering products and services to differentiate and enhance the user experience of portable telephone devices. TAT's business mainly focuses on graphical user interfaces (GUI). This report is a planning of a web-based educational material for TATs XML-based programming languageTAT Cascades.

TAT Cascades is built on the Kastor platform, and together these two programming languages create graphical user interfaces for mobile telephones. These interfaces are in general more dynamic, faster and richer in a multimedia experienced way than other mobile phone interfaces. A short declaration of these two languages would be that Cascades handles the communication between components (like lists, menus) and Kastor acts as their graphical renderer. This means that a part of Cascades consist of Kastor, but both of these programming languages are XML-based and for the unaccustomed user they can be hard to separate.

During the thesis a webpage which contains tutorials for these two programming languages has been created. On this page it will be declared how the file structure for different GUIs is created, connections between components are sorted out, and intricate code elements are explained. After the code has been mapped it's possible to look at, and run the code through the belonging editor that's being downloaded via the homepage. The code is simulated by a simulator that's run with a button in the editor, and presents the user with a graphical representation of what this particular code would look like in a mobile telephone. In the end of each tutorial there will be exercises. To make the learning experience more interactive these exercises can be solved in the editor.

The simulator that's being run by the editor is a tool developed by TAT and it was available before we started the thesis. The simulator won't be discussed any further in this report.

A prototype of the webpage, and editor has been created during the thesis. It can be found on:
http://www.jonathan-stahl.se/1EXJOBB/login.php

Requests for login, and passwords can be sent to:
Ihm04pno@student.lu.se or jonathan.stahl.410@student.lu.se

Keywords: Tutorial, education, interactivity, PHP, SQL, JavaScript, Cascades, Kastor, editor

Förord

Idén att göra examensarbetet på TAT kom ganska plötsligt när vi snubblade in på TATs hemsida. Där blev vi minst sagt mycket inspirerade av vad vi såg. Att arbeta med teknik och design på detta sätt verkade verkligen trevligt. När arbetet kommit i gång fortskred dom första 6 veckorna på TATs kontor i Malmö och resterande tid bestod av arbete i skolan och hemma. Vi vill tacka TAT för möjligheten att göra examensarbetet hos dem samt vår handledare Daniel Johansson på TAT som vi fört ett nära samarbete med. Ett tack även till vår handledare på skolan, Mats Rüder.

Innehållsförteckning

1 Inledning 1
1.1 Bakgrund 1
1.1.1 Om TAT 1
1.1.2 Om Kastor och Cascades 1
1.2 Nuvarande behov på TAT och deras lösningsförslag 1
1.2.1 Bristfällig Dokumentation 1
1.2.2 Avsaknad av övningar 2
1.2.3 Komplexa GUI:s att börja inlärningen med. 2
1.2.4 Dynamiskt innehåll. 3
1.3 TATs kravspecifikation 3
1-Funktionella krav, hemsida 3
2-Kvalitetskrav, hemsida 4
3- Funktionella krav, editor 4
4- Kvalitetskrav, editor 4
1.4 Problembeskrivning 4
1.4.1 Syfte 4
1.4.2 Mål 4
1.4.3 Frågeställningar 4
1.4.4 Målgrupp 5
1.4.5 Avgränsningar 5
2 Metodik 6
3 Genomförande 8
3.1 Granskning av TATs kravspecifikation 8
3.1.1 Krav på hemsidan 8
3.1.2 Krav på editorn 9
3.2 Grafisk design av hemsida och editor 12
3.2.1 Hemsidans inloggningssida 12
Login-sidan 13
3.2.2 Hemsidans huvudsida 14
Vanligt användar-konto 15
Användarkonto med administratörsrättigheter 16
3.2.3 Editorns presentation av övningar 18
3.2.4 Editorns hantering av kod 19
3.3 Programmeringsspråk till hemsida och editor 22
3.3.1 Hemsida – HTML (HyperText Markup Language) 22
3.3.2 Hemsida – PHP 22
3.3.3 Hemsida – JavaScript 23
3.3.4 Hemsida – SQL (Structured Query Language) 23
3.3.5 Hemsida – CSS (Cascading Style Sheets) 24
3.3.6 HTML Applications 24

3.3.7 Macromedia Director 25
3.3.8 Java 26
3.3.9 JavaScript (Editor) 27
3.3.10 C++ 27
3.3.11 .NET 28
3.3.12 Webbsida 29
3.4 Skapande av testmiljö 31
3.5 Implementering 32
3.5.1 Grafiskt gränssnitt, hemsida: 32
3.5.2 Teknisk utveckling, hemsida: 32
3.5.3 Flödesschema av blockfunktionerna på hemsidan: 40
3.5.4 Grafiskt gränssnitt, editor: 42
3.5.5 Katalogstruktur, editor: 42
3.5.6 Teknisk utveckling, editor: 42
3.5.7 Lägga till övning till editorn 46
3.5.8 Problem under utveckling av editorn och lösning. 47
4 Resultat och Analys 50
4.1 Resultat 50
4.2 Granskning av kravspecifikation 50
4.3 Granskning av arbetsmetodiken 50
4.4 Granskning av implementering och alternativa lösningar 51
4.5 Erfarenheter från examensarbetet 52
4.6 Kommentarer på applikationen från nyanställd på TAT 52
5 Slutsats 54
Appendix A – TAT Tutorials kravspecifikation 55
1-Introduktion 55
2-Krav på hemsidan 55
3- Krav på editorn 56
Appendix B - Ordlista 58
Appendix C – Kodfil till editorn 59

1 Inledning

1.1 Bakgrund

1.1.1 Om TAT

TAT grundades 2002 och är ett företag som skapar användargränssnitt (GUI) för mobil teknik. TATs huvudkontor ligger i Malmö och har sedan starten 2002 varit mycket framgångsrikt inom detta område. Hittills har dem fördubblat sin omsättning för varje år som går. TAT skapar sina GUIn med hjälp av två XML-baserade programmeringsspråk. Dessa heter Kastor och Cascades och har utvecklats av företaget.

1.1.2 Om Kastor och Cascades

I denna rapport kommer programmeringsspråken Kastor och Cascades namn att frekvent förekomma. Därför skall vi ge en snabb och övergripande bild av vad dessa XML-baserade språk egentligen är.

TAT Cascades är ett ramverk av komponenter som ger designers nya verktyg för att skapa gränssnitt. Dessa komponenter kan vara menyer, popup menyer, listor osv. Cascades hanterar dessa komponenter och kommunikationen mellan dem. Det visuella uppträdandet av alla UI komponenter är lätt att skräddarsy eftersom språket bygger på XML standarder.

TAT Kastor är renderaren till Cascades. Det är en plattformsoberoende grafikmaskin för att presentera gränssnitt. Det är designat för att hantera användarkrav på grafik, animationer och övergångar mellan olika skärmbilder.

Detta betyder att Cascades inte kan stå utan Kastor. Antingen köper en kund Kastor och applicerar på deras egen modell, eller Kastor och Cascades som ett helt paket. Båda programmeringsspråk är likt html, tag-baserade och för en ovan användare är dem svåra att skilja.

1.2 Nuvarande behov på TAT och deras lösningsförslag

1.2.1 Bristfällig Dokumentation

Större delen av den nuvarande dokumentationen om Cascades listar endast olika element, dess attribut och något exempel till de flesta av dessa element. Detta är ett bra komplement till någon som redan är insatt i Cascades, men fungerar inte så bra om den ska stå på egna fötter som enda läromedel.

Dokumentationen saknar grafiska exempel till kodavsnitten vilket gör det svårt att visualisera resultaten och verkligen se vad som händer.

Lösning: Hemsida

Istället för att ha ett stort antal papper som förklarar programmeringsspråken, så vill TAT presentera informationen på en hemsida där det finns handledningstillfällen. Handledningstillfällena följer uppbyggnaden kapitelvis, i skapandet av olika GUI:n. Samtidigt som detta görs så förklaras vad koden i de filer som behöver skapas gör och varför den behövs på ett betydligt mer grundläggande sätt än i dokumentationen. För varje nytt handledningstillfälle finns även beskrivningar om vilka filer som är nya och var de ligger. Genom att beskriva var filerna ligger på detta sätt med både ord och bild, så lär sig alla skapa samma katalogstruktur. Detta gör det mycket lättare för alla om man senare behöver arbeta i GUI:n skapade av andra personer. Endast anställda och kunder till TAT ska ha möjlighet att lära sig Cascades på denna sida.

1.2.2 Avsaknad av övningar

I nuläget så har nya användare av Cascades fått testa sig fram genom att ändra värden och kod i befintliga GUI:s. TAT vill att det ska finnas övningar som ger användarna möjlighet att bygga upp ett komplett GUI från grunden steg för steg.
Dessa övningar med instruktioner ska finnas till varje kursavsnitt på hemsidan.

Lösning: Editor

Att inte få interagera och laborera med kod, att inte få testa och se om de saker man förväntade sig skulle hända verkligen händer, är en stor brist i inlärningsprocessen enligt TAT. Därför ska en editor skapas där man kan lösa uppgifter som ges i slutet av varje kursavsnitt på hemsidan. I handledningstillfällena på hemsidan går man igenom kod och förklarar vad denna gör, även denna kod skall vara möjlig att köra i editorn.

1.2.3 Komplexa GUI:s att börja inlärningen med.

I nuläget så har nyanställda på TAT studerat gamla exempel på olika GUI:s för att lära sig Cascades. Detta gjordes i kombination med att läsa pappersdokumentationen. Detta är inte så effektivt av olika anledningar. Bara ett simpelt GUI kan bli upp till hundra filer och fler. Detta gör det besvärligt för en ovan Cascades kodare att leta sig fram till en speciell fil som berör en viss data eller visuell funktion. Att testa sig fram i ett färdigt GUI är en bra idé men att hitta i mappstrukturen och förstå sig på hur alltsammans hänger ihop är svårt då allt är färdigt och man inte får bygga upp något själv.

Lösning: Snabb-knappar till övningsspecifika filer i editor

Istället för att bläddra genom stora mängder filer som finns i ett GUI så ska en förenklad editor användas där endast de viktigaste filerna för kursavsnittet ska synas. Detta gör det enklare för användaren att se var ändringar behöver göras för att påverka vissa effekter i GUIt.

1.2.4 Dynamiskt innehåll.
Cascades är inte färdigutvecklat och förändras därför mycket. Nya funktioner kommer att dyka upp och äldre förändras. Detta kräver att dokumentation uppdateras vilket kan vara komplicerat om allt ligger i ett textdokument som senare måste skickas ut på nytt till alla användare. Detta innebär också att gamla exempel-GUI:n kan baseras på äldre simulatorer, dvs. gamla Cascades versioner och att dokumentationen inte går hand i hand med det demo man studerar. Med hand i hand menas att attributen som förklaras till ett element kan vara helt andra än de man har i sin egen fil och att se samband med de attribut som finns i exemplet kan vara svårt eftersom dokumentationen går igenom helt andra attribut.

Lösning: Dynamisk hemsida och editor
Innehållet på hemsidan ska gå att uppdatera av en anställd på TAT och existerande övningarna till editorn ska kunna förändras. Nya övningar ska kunna läggas till och gamla ska kunna tas bort.

Det är dessa behov som satt huvudriktlinjerna för de krav som TAT har ställt till oss. Här nedan kommer deras kravspecifikation i sin helhet.

1.3 TATs kravspecifikation

Här nedan är en lista på krav som TAT kommit fram till efter diskussioner sinsemellan och med oss. Dessa krav baseras på problemen med deras nuvarande situation.
Kraven är väldigt generella och det är meningen att vi ska utveckla dem samt göra nödvändiga kompletteringar senare i arbetet.

1-Funktionella krav, hemsida

1.1- För att editorn skall gå att använda så behövs ett kompletterande ramverk, detta ramverk görs i form av en hemsida.

1.2- På hemsidan finns handledningstillfällen.

1.2.1- Varje handledningstillfälle är uppdelat i sektioner som kallas kapitel.

1.3- Hemsidan skall använda sig av ett inloggningssystem så att obehöriga ej kan ta del av utbildningsmaterialet.

1.3.1- Inloggningssystemet skall ha användarnamn och lösenord.

1.3.2- Det skall finnas administratörskonton och användarkonton.

1.3.3- Endast administratörskonton kan förändra innehåll på hemsidan.

2-Kvalitetskrav, hemsida

2.1- Språket på hemsidan skall vara engelska.

3- Funktionella krav, editor

3.1- Editorn ska innehålla övningar som motsvarar alla kapitel som finns på hemsidan.

3.2- Editorn ska ha minst en kodfil att visa upp för varje övning.

3.2.1- Kodfilernas kod ska gå att återställa när man vill, till sitt ursprungliga skick med hjälp av en Reset-knapp.

3.2.2- Koden ska gå att testköra när som helst med hjälp av ett tryck på Run Simulator-knappen.

3.3- Editorn ska färgkoda koden i filerna enligt XML standard.

3.3.1- Färgkodningen ska ske i realtid.

3.4- I editorn skall det finnas en uppdatera knapp, på vilken man kan kontrollera mot en server om det finns en nyare version av editorn.

4- Kvalitetskrav, editor

4.1- Det ska finnas en hjälpfil till editorn som förklarar grunderna i användandet av editorn.

4.2- Språket i editorn ska vara engelska.

1.4 Problembeskrivning

1.4.1 Syfte

Syftet med projektet är att skapa en applikation som underlättar för Cascades-nybörjare att lära sig detta programmeringsspråk.

1.4.2 Mål

Målet är att skapa ett läromedel i form av en hemsida med tillhörande editor till TATs XML-baserade programmeringsspråk TAT Cascades. På hemsidan kommer handledningstillfällen att ges tillsammans med deluppgifter som kan lösas i editorn. Endast ramverken till applikationerna ska skapas, ej deras innehåll.

1.4.3 Frågeställningar

Vilka är TATs krav och hur kan vi komplettera dem för att bäst uppfylla TATs behov?

Vi kommer att studera TATs krav samtidigt som vi har arbetets syfte, mål och TATs egentliga behov i tankarna. Då kommer en stor mängd frågor att uppstå som måste besvaras innan arbetet kan genomföras. Dessa frågor kommer att besvaras och sedan sammanställs en komplett kravspecifikation (se avnsitt 3.1 Granskning av TATs kravspecifikation).

Hur ska hemsida och editor byggas upp och utformas?
Hur ska informationen presenteras på hemsidan? Ska editorn vara inbyggd på hemsidan eller ligga separat? Vilka plattformar passar oss bäst för att utveckla dem?
Dessa frågor måste besvaras samt frågor angående editorns och hemsidans funktionella design.

1.4.4 Målgrupp
Målgruppen är nyanställda på TAT samt kunder som köpt rättigheterna att använda Cascades och Kastor av TAT. Dessa användare har en robust programmeringsgrund att stå på. På vilket sätt kommer detta att påverka utformandet av vår applikation?

1.4.5 Avgränsningar
Eftersom TATs simulator till Cascades endast fungerar i Windows miljö kommer vi att fokusera oss på att skapa en hemsida och editor som fungerar på PC datorer med Windows som operativsystem. Någon vetenskaplig undersökning angående pedagogiken i applikationen kommer inte att göras eftersom det skulle ta för mycket tid och resurser.
Ingen större tyngd kommer att läggas på det estetiska arbetet. Detta eftersom TAT med all sannolikhet senare kommer att göra om utseendet på applikationen så att den passar in i deras grafiska mall.

2 Metodik

2.1 Granskning av Kravspecifikationen

För att skapa oss en bild av vad det är vi ska utforma så kommer vi att studera kravspecifikationen som TAT skrivit. Vi kommer att presentera en detaljerad förklaring av alla krav, varför dem finns och vad dem innebär. Under tiden som vi studerar TATs kravspecifikation så görs eventuella kompletteringar som vi tycker är nödvändiga för att skapa en bättre funktionalitet. Kraven som läggs till listas och vi motiverar varför vi väljer att ligga till de krav som läggs till. Slutgiltig kravspecifikation finns sedan att läsa i appendix.

2.2 Skapande av sketch till grafisk design och visuella funktioner

Efter att vi har studerat och kompletterat TATs kravspecifikation så kommer en sketch av hur informationen grafiskt skall presenteras för användaren att skapas. Kraven och handledningstillfällena som TAT skrivit ställer indirekt krav på hur applikationen behöver se ut. Vi funderar på hur dessa skulle kunna representeras grafiskt och skapar en sketch som vi sedan kommer att använda när applikationen skapas under implementeringen. När sketchen är gjord presenteras denna och utifrån resultatet motiverar vi varför vi gjort dem designval som gjorts. Om där finns designval som skapar nya funktioner för att förbättra funktionaliteten av applikationen så kommer vi att motivera dessa.

2.3 Val av programmeringsspråk

Efter att ha studerat skapat designsketchen och vet precis hur applikationen ska gestalta sig så kommer vi att besluta vilka programmeringsspråk som kan implementera de funktionella och designmässiga krav som blivit ställts på hemsidan och editorn. Vi väger för- och nackdelar med de olika programmeringsspråken mot varandra. Först kommer de programmeringsspråk vi är vana vid och kan hantera bra att undersökas, efter detta undersöks hur pass bra alternativa programmeringsspråk skulle fungera. De programmeringsspråk som lämpar sig bäst att utveckla applikationen i kommer att användas

2.4 Skapa testmiljö

Efter att ha valt programmeringsspråk så ska vi skapa en testmiljö. Testmiljön skapas för att kravspecifikationen kräver en databas för att lagra det dynamiska innehåll som ska finnas på hemsidan. Databasen kommer att ligga på en separat server för att undvika att skapa problem med målservern.

2.5 Implementering

Genom att använda de programmeringsspråk vi valt i tidigare så implementeras hemsidan och editorn enligt den sketch som skapats i design kapitlet. Informationen som ska presenteras på hemsidan kommer från databasen, upprättad i testmiljön. Vi kommer att utföra kontinuerliga tester parallellt med

utvecklingen av hemsidan och editorn för att upptäcka eventuella fel som uppstår. När implementeringen är klar så ska TAT ha en hemsida och editor vilka kan användas som utlärningsmedel för deras programmeringsspråk Cascades.

2.6 Analys av resultat

Genom att undersöka och analysera resultatet och tillvägagångssättet kan eventuella misstag eller alternativa lösningar listas. Vi kontrollerar om vårat resultat uppfyllde alla de krav ställda i kravspecifikationen. Om inte, varför blev det fel och hur kunde det ha undvikits.

3 Genomförande

3.1 Granskning av TATs kravspecifikation

För att vi skulle få en klar bild av vad som skulle utformas så studerade vi först kravspecifikationen given av TAT. I denna kravspecifikation gjorde vi sedan eventuella kompletteringar. Detta kapitel går igenom TATs krav punktvis. Vi förklarar TATs krav mer ingående. Vi motiverar även här varför vissa krav behöver kompletteras. Dessa kompletterande krav tar vi även upp här. Den färdiga kravspecifikationen med både TATs krav och våra kompletterande krav finns att se i appendix A.

3.1.1 Krav på hemsidan

TATs krav på en hemsida med handledningstillfällen[1]:
TAT vill att deras dokumentation ska presenteras på en hemsida. Denna dokumentation är uppdelad i olika handledningstillfällen som alla består av underkapitel. Dessa underkapitel innehåller alla mindre paragrafer, vissa bara text, andra med tillhörande bilder och vissa består av stycken med kod.

TATs krav på att hemsidan ska ha ett inloggningssystem[2]:
Vi ska skapa ett inloggningssystem där ett loginnamn och ett lösenord kommer att krävas av användaren. Detta av den enkla anledningen att vem som helst inte får ta del av informationen som står på hemsidan. Vi kommer att dela upp kontona i två grupper. De med skrivrättigheter och de utan.

TATs krav om att endast administratörer ska kunna ändra innehållet på hemsidan[3]:
TATs krav är att en användare med skrivrättigheter (dvs administratör) ska kunna ändra innehållet på hemsidan.

Detta eftersom det enkelt skall gå att uppdatera hemsidan direkt via webben. De konton som har skrivrättigheter kan skapa, uppdatera och ta bort information från handledningstillfällena. Konton utan dessa rättigheter kan endast läsa information.

TATs krav på att hemsidan ska vara på engelska[4]:
TAT har ett krav på att all text på hemsidan ska vara på engelska. Anledningen är att deras kunder och anställda kommer från olika delar av världen och att engelska är det mest tillgängliga språket.

---

[1] Detta stycke berör TATs krav 2.1, 2.2.
[2] Detta stycke berör TATs krav 2.3,2.3.1, 2.3.2.
[3] Detta stycke berör TATs krav 2.3,3.
[4] Detta stycke berör TATs krav 2.5.

Våra kompletterande krav till hemsidan[5]:
Vi tyckte att vissa krav behövdes läggas till för att begränsa hemsidan vad gäller design och maximera tillgängligheten av hemsidan.

Hemsidans utseende ska anpassas efter upplösningarna 1024*768 och större med tanke på att alla användare ska kunna se allt på sidan utan problem. Upplösningen 800*600 hade vi tänkt oss först men med tanke på den mängd kod som ska visas och den bredd vissa kodrader har så uteslöts den.

Hemsidan kommer att innehålla en del bilder och kodstycken vilket kan bli lite tungt för en modem-användare. Självklart så ska vi försöka optimera hemsidan så mycket som möjligt och har satt ett krav på att användare med minst 512 kbit/sekund ska kunna besöka sidan utan problem.

Eftersom hemsidor inte ser exakt likadana ut i olika webbläsare så har vi valt att optimera hemsidan för Microsofts Internet Explorer med tanke på att det är den vanligaste. Detta innebär inte att den inte kommer att fungera i andra webbläsare som Mozilla Firefox, men den kommer kanske inte se lika bra ut.

3.1.2 Krav på editorn

TATs krav om att editorn ska vara en fristående applikation från hemsidan[6]:
TAT vill att editorn ska vara en separat applikation från hemsidan. Det vill säga att den inte ska vara inbyggd på hemsidan. Detta för att de tycker det är viktigt att visningen av koden i övningsfilerna är så tydlig som möjlig. Kodraderna blir ofta väldigt breda vilket kräver ibland att man måste maximera editorns fönster för att se dem bra. Att bygga in editorn på hemsidan hade också försvårat åtkomsten till information på hemsidan samtidigt som man arbetade i editorn. Detta eftersom man då måste öppna ett nytt webbläsar-fönster och bläddra sig fram till handledningstillfällena igen.

TATs krav om att editorn ska färgkoda koden i realtid[7]:
TAT har krav på att all kod man skriver i editorn ska färgkodas i realtid enligt XML-standard. För att lättare kunna urskilja olika element i kod så brukar många programmeringseditorer använda sig utav färgkodning. Ett exempel på färgkodning är följande:

```
<container id=”window” width=”100” height=”300”>
```

[5] De krav som lagts till som berör detta kan ses i den kompletta kravspecifikationen i appendix A (se punkterna 2.5, 2.6, 2.7).
[6] Detta stycke berör TATs krav 3.1.
[7] Detta stycke berör TATs krav 3.2, 3.2.1.

Det gör det enklare att skilja olika element från dess attribut och attributens värden.
Att färgkodningen ska ske i realtid innebär att när användaren knappar in kod så kommer färgkodningen att ske direkt och texten som just skrivits kommer att ändra färg. TATs krav är att koden ska färgkodas enligt XML-standard, detta på grund av att deras programmeringsspråk är baserat på XML-språk och har samma typ av struktur som XML.

TATs krav om att editorn ska ha samma övningar som visas på hemsidan[8]:
TAT har ett krav om att alla övningar som finns på hemsidan ska finnas tillgängliga i editorn. Detta ska göra det möjligt för en användare att använda sig utav endast editorn utan hemsidans hjälp om denne känner för att bara göra övningar och kan tillräckligt för att göra dem utan hemsidans hjälp.

TATs krav om att editorn ska ha minst en kodfil per övning och att koden i den ska gå att återställas[9]:
TAT har ett krav på att varje övning i editorn ska ha minst en kodfil att visa upp. Det är inte vi som ska göra övningarna men detta krav innebär att vi måste lösa hur dessa kodfiler ska visas upp och växlas mellan om det finns flera. TATs krav om att kunna återställa koden i en kodfil är egentligen en av grundtankarna med editorn. En återställnings-knapp ska finnas tillgänglig i editorn som möjliggör total återställning av koden i en övning. Detta med tanke på att en användare kan göra misstag och förstöra koden så att ingenting fungerar längre. Om man då trycker på återställnings-knappen så återställs koden i alla filer för övningen man håller på med.

TATs krav om att editorns kodfiler ska gå att köras i deras simulator[10]:
TATs krav om att kunna testköra koden i deras simulator är en väldig viktig funktion. Detta ger nämligen användaren möjlighet att se resultat på sina ändringar i koden. Det ska gå att starta igång simulatorn från editorn. Koden man arbetar med ska laddas in automatiskt till simulatorn så att det snabbt och enkelt går att se resultat av ens kodning.

TATs krav om att det ska vara enkelt att lägga till övningar till editorn[11]:
Enligt TATs krav så ska man enkelt kunna lägga till övningar till editorn. Detta kräver en djupare förståelse för hur TATs simulator fungerar med tanke på att denna ska hitta alla filer till de olika övningarna för att kunna köra dem. En lösning på detta problem kommer vi fram till i implementeringsdelen av denna rapport.

[8] Detta stycke berör TATs krav 3.3.
[9] Detta stycke berör TATs krav 3.5, 3.5.1.
[10] Detta stycke berör TATs krav 3.6.
[11] Detta stycke berör TATs krav 3.7.

TATs krav om att alla instruktioner och text ska vara på engelska[12]:
Ett av TATs krav på editorn är att all text ska vara på engelska, precis som med hemsidan med tanke på deras kunder och anställda kommer från många olika nationaliteter som förklarats tidigare.

Utöver det kravet så vill de också att en hjälpfil ska finnas tillgänglig för användaren om hur man ska använda editorn. Denna hjälpfil ska förklara kortfattat hur man använder de olika knapparna och funktionera i editorn.

Våra kompletterande krav till editorn[13]:
Att editorn ska vara en fristående applikation från hemsidan är ett krav från TAT. Men detta kände vi innebar det underförstådda kravet om att editorn ska vara tillgänglig för nerladdning på hemsidan.

TATs krav om att alla övningar på hemsidan ska finnas tillgängliga i editorn kompletterade vi också. En övning kommer att finnas i slutet av varje kapitel på hemsidan och dessa övningar ska ha samma namn i editorn som kapitelnamnen på hemsidan. Detta tror vi ska göra det enklare för användaren att hitta rätt övning till motsvarande kapitel på hemsidan. En annan anledning är att alla kapitlen har namn som beskriver vad de innehåller vilket gör det enklare för användaren att se vad övningen handlar om.

Ett krav som vi inte fick av TAT men som vi kände skulle behövas är möjligheten att uppdatera editorn. Detta kan vi kanske lösa genom att lägga till en uppdatera-knapp som sedan kollar med en server om en senare version av editorn finns.

Angående TATs krav om återställnings-knapp som ska möjligöra total återställning av koden i en övning så la vi till ett kompletterande krav. Här tyckte vi att vi kunde komplettera med ett krav om en återställnings-knapp som återställer koden i endast den fil man håller på att arbeta i. Anledningen till detta är att om man är nöjd med ändringarna i vissa filer men vill återställa koden i endast en fil så ska man kunna återställa koden i denna fil utan att behöva göra om det som man var nöjd med i de andra.

Vi har även kompletterat TATs krav om en hjälpfil som förklarar hur man ska använda editorn. Vår komplettering innebär ytterligare information till TATs anställda som ska förklara hur man ska göra för att lägga till en övning till editorn.

[12] Detta stycke berör TATs krav 3.8.
[13] De krav som lagts till kan ses i den kompletta kravspecifikationen i appendix A (se punkterna 3.1.1, 3.3.1, 3.4, 3.9.1, 3.10).

Ett sista krav vi kände behövdes för editorn var angående dess filstorlek. Detta med tanke på att den ska laddas ner från hemsidan. Om dessutom editorn uppdateras ofta med nya övningar så är det viktigt att inte filstorleken blir för stor om användarna måste ladda ner den på nytt fler gånger.

3.2 Grafisk design av hemsida och editor

I TATs kravspecifikation så innebär vissa funktionaliteter ett designmässigt beslut. Dessa beslut har vi tagit utifrån hur vi tycker att applikationen utformar sig bäst, och motiveringar till olika designval ges. Vi kommer dock inte att lägga någon större tid på att få applikationen att se visuellt tilltalande ut. Detta på grund av att TAT själva vill att vi ska koncentrera oss på funktionaliteten på applikationen och inte dess utseende. Detta har de påpekat genom att säga att möjligheten finns att den inte kommer att användas i den form vi skapar den. Hemsidan kommer förmodligen att byggas in på den befintliga hemsida som finns på deras intranät och editorn kommer att anpassas till deras grafiska mall. Vi ska försöka göra vår design så att den blir bra ur ett pedagogiskt perspektiv. Men vi vårt arbete lägger större vikt på det tekniska, funktionsinriktade aspekterna av hemsidan och editorn och vi tar inte hänsyn till huruvida applikationen blir grafiskt tilltalande eller pedagogiskt väl utformad.

Hemsidan:

Enligt vår tolkning av TATs kravspecifikation så vill vi utforma hemsidan på sådant sätt att den har en inloggningssida, en meny för att komma åt alla handledningstillfällen och ett utrymme för brödtext, där all information ska visas, så som text, kod och bilder. Vårt beslut om detta är inspirerad av de hemsidor vi själva besökt och använt oss utav för att lära oss olika programmeringsspråk under vår tid på utbildningen och fritid. Inloggningssidan ska vi skapa för att uppfylla TATs krav om inloggningssystem, denna sida har vi tänkt ska också fungera som startsida för hemsidan. Det viktigaste på hemsidan förutom dess innehåll är enligt oss hur navigering kommer att ske, och att den är enkel för användaren att utföra. Användaren ska lätt se var hon är på sidan, vart hon kan gå vidare och hur hon ska göra för att komma dit hon vill.

3.2.1 Hemsidans inloggningssida

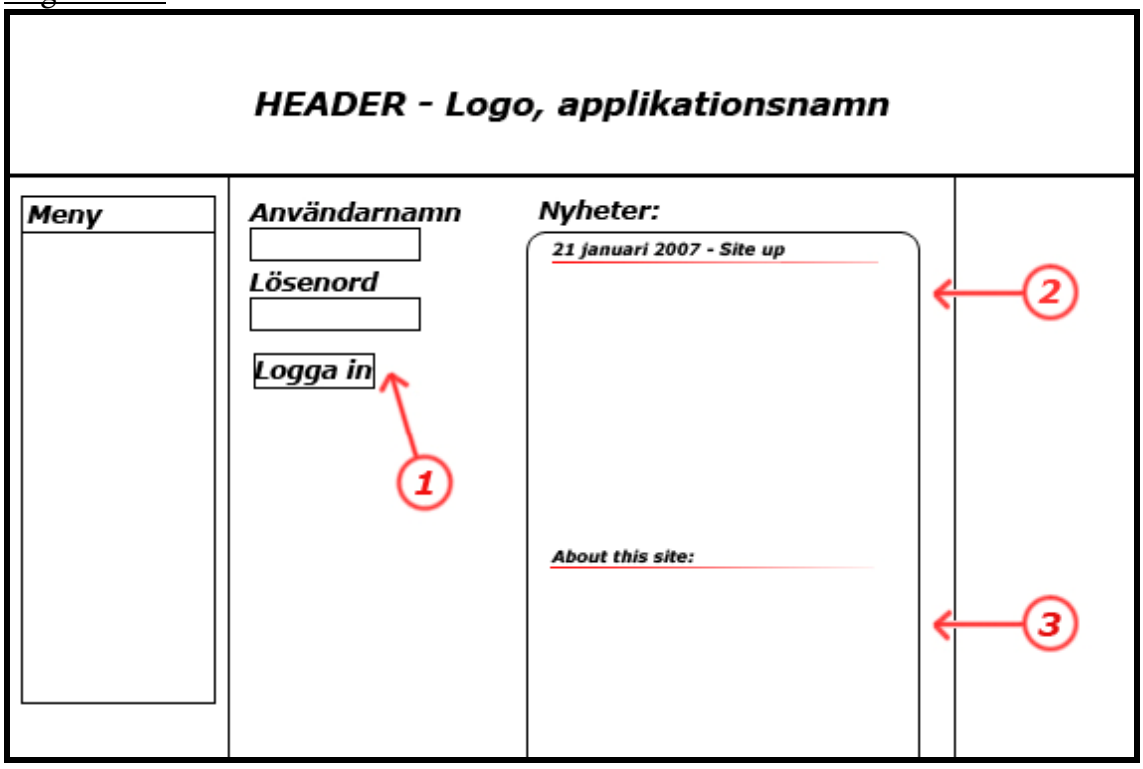
Enligt TATs krav så behövs en inloggningssida eftersom där ska finnas två typer av användarkonton. Ett konto med läs och skrivrättigheter (även kallad administratör) och ett vanligt användarkonto med endast läsrättigheter.

Ett vanligt inloggningssystem brukar bestå utav två inmatningsfält där det ena kräver en identifikation i form av namn eller loginnamn. Detta kan vara användarens riktiga namn eller bara ett ID. Det andra fältet kräver ett lösenord som bekräftar att det verkligen är personen i fråga som vill få åtkomst till sidan. Till slut krävs det en knapp som användaren måste trycka på efter att fyllt i de

båda fälten. När knappen trycks ner skickas informationen från fälten till en server som kontrollerar om den inmatade texten stämmer överens med det data som finns i databasen. Gör den det, så skickas användaren vidare till sidan med handledningstillfällen eller om inte så genereras ett felmeddelande.

Här nedan är en skiss på hur vår loginsida ska se ut. Menyn som syns på loginsidan kommer inte vara tillgänglig förrän man har loggat in. En yta till höger om loginfälten kan visa nyheter om sidan och annan information. Loggan skall vara uppe till vänster eftersom vi tror det är här västerländska användare omedvetet tittar och lägger märke till först med tanke på att vi läser från vänster till höger, uppifrån och ner. Efter loggan ska titeln på själva hemsidan stå. Denna ”header” kommer att följa med och vara samma på alla sidor. Detta eftersom användare alltid ska veta var de befinner sig och vilket företag hemsidan presenteras av.

Login-sidan



1- **Inloggningsfält:**
Här skriver man in användarnamnet och lösenordet som man fått av TAT då man köpt deras produkt Cascades. Trycker man sedan på knappen ”Logga in” kommer man in på huvudsidan och får åtkomst till handledningsavsnitten. Här loggar man in både som vanliga användare och som användare med administratörsrättigheter. För att lätt kunna se inloggningsfältet så ligger det uppe till vänster utan andra störande grafiska element runt sig. Detta för att ingen användare ska behöva leta efter login

fältet, det ska synas direkt och vara bland det första användarens ögon fastnar på.

2- **Nyheter:**
Här finns utrymme för ett nyhetssystem. Nyhetssystemet går inte att uppdatera via hemsidan eftersom man inte är inloggad. Nyhetssystemet kan vara bra då man vill uppmärksamma användare om ändringar, nyheter och annat som kan vara av intresse för den vanlige användaren.

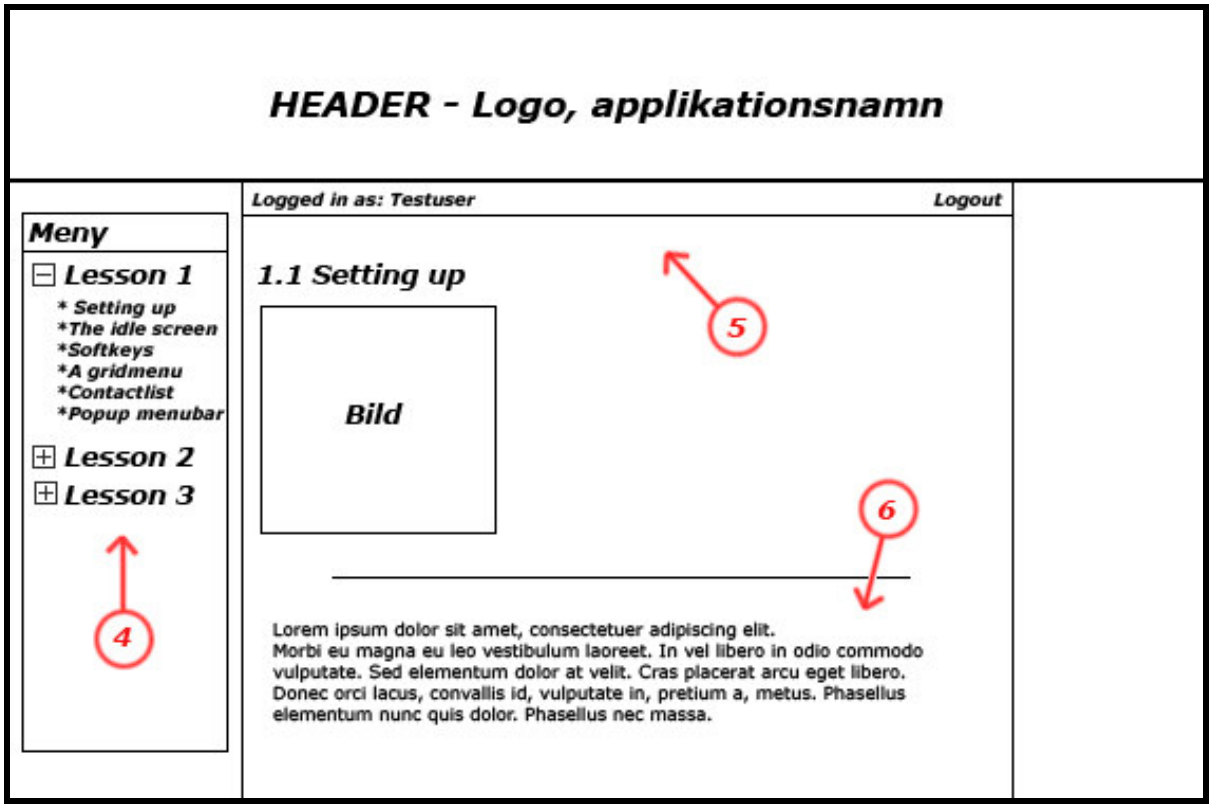
3- **Information om hemsidan och TATs Cascades:**
Här visas information och instruktioner för förstagångsanvändaren om Cascades samt TAT Tutorials.

3.2.2 Hemsidans huvudsida

Hemsidans huvudsida är där all brödtext visas, alltså allt innehåll som kommer att visas på hemsidan i form av handledningstillfällen och övningar.
Den text som kommer att visas på hemsidan ska bestå av handledningstillfällen som redan är skrivna av TAT.

Här följer två skisser på huvudsidan. Den första bilden visar utseendet på sidan för en användare med skrivrättigheter. Den andra bilden representerar sidans utseende för en användare utan skrivrättigheter.
På varje skiss finns en pil med en siffra. Dessa siffror motsvarar olika områden på hemsidan som alla har en motsvarande förklaring under bilden.

Vanligt användar-konto



4- **Meny:**
Här visas alla kursavsnitt på hemsidan, uppdelade i lektioner och om man trycker på dessa, dess underkapitel. De fungerar som länkar till varje kursavsnitt. Menyn ska gå att expandera och minimera om det visar sig bli många olika kapitel. Menyn ligger till vänster och påverkar innehåll som finns i den stora yta i mitten som är skapad för att presentera innehåll. Detta för att vi tror användare är vana vid detta och känner sig igen sig i.

5- **Inloggningsinformation:**
Här visas vem man är inloggad som, samt en knapp för att logga ut.

6- **Brödtext:**
Här visas information som finns i ett underkapitel till ett kursavsnitt. Det vill säga alla dokumentation TAT har skrivit och övningar som sedan ska lösas i editorn.

Användarkonto med administratörsrättigheter

TATs handledningstillfällen är uppdelade i paragrafer som antingen innehåller text, bild eller kod. Detta tolkar vi som att innehållet på hemsidan kommer att bestå utav tre olika typer av block: textblock, bildblock och kodblock. Att tänka på innehållet som flera mindre block skulle kunna göra redigering av innehållet enklare för användare med administratörs konton. Detta på grund av att de skulle slippa redigera ett helt handledningstillfälle på en gång utan kan redigera det styckvis. Antingen redigerar de ett textstycke utan att påverka bild och kod, eller byter ut en bild utan att påverka resten av innehållet och samma sak med koden. Att texten dessutom hamnar blockvis på detta sätt ser mer harmoniskt ut och känns inte lika tungt att stega igenom som om allt innehåll skulle stått uppradat i ett enda långt stycke tror vi.

När en anställd på TAT med administratörsrättigheter loggar in på hemsidan så kommer det att dyka upp tre små knappar(ikoner) intill varje block.

En knapp för att lägga till ett nytt block (den gröna), en knapp för redigering av blockets innehåll (den gula) och en knapp för att ta bort blocket (den röda):

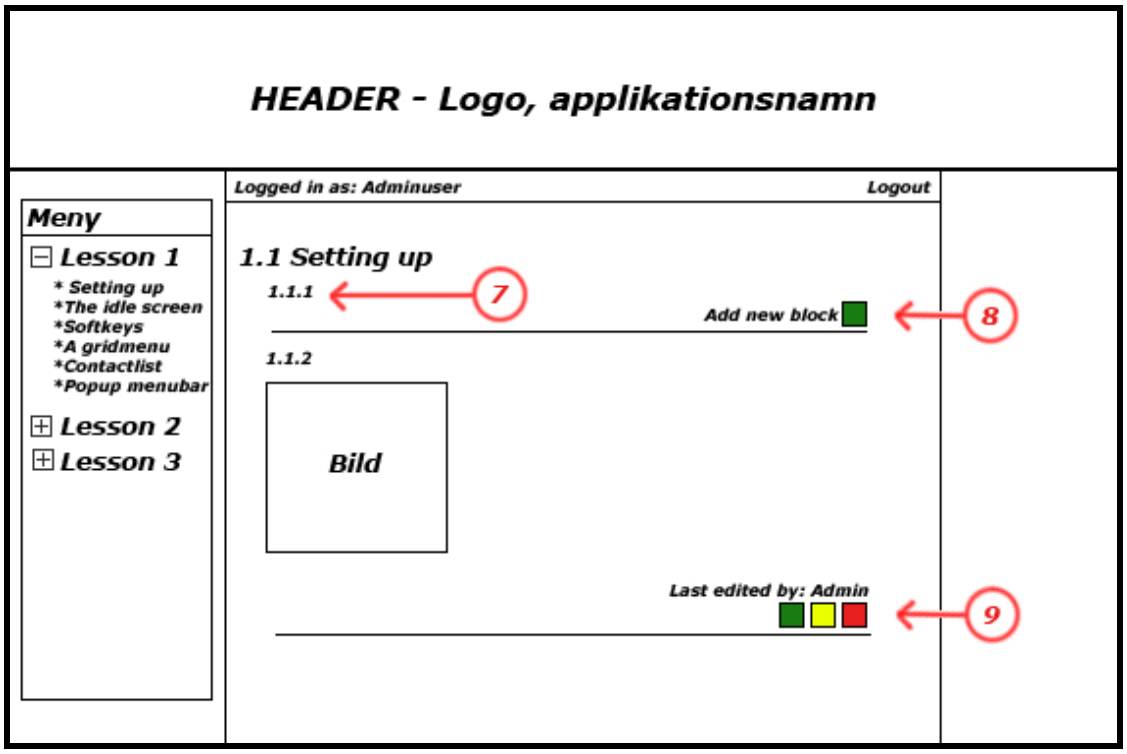
- Väljer man att lägga till ett block efter det aktuella blocket så ska en popup dyka upp. Denna har de tre olika blocktyperna listade. Här väljer användaren om hon vill lägga till text, bild eller kodblock. Dessa tre blocktyper kommer att ha alla att ha varsin motsvarande ikon.

- Om en användare trycker på redigera-knappen till ett särkilt block så ska ett textfält med tillhörande blocks innehåll i dyka upp. Där kan administratören ändra innehållet och sedan trycka på en acceptera-knapp som ändrar blockets innehåll till det förändrade innehållet i textfältet.

- Skulle användaren välja att ta bort blocket så ska det visas en popup bredvid knappen som frågar om det är säkert att hon vill ta bort blocket.

Dessa tre knappar kommer bara att synas för användare som är inloggade med administratörsrättigheter. Under dessa tre knappar till varje block ska det stå utskrivet vilken administratör som senast redigerat tillhörande block.
Detta blir praktiskt då en administratör enkelt kan se vem som bär ansvaret för en viss redigering om en förklaring behövs. Knapparna ligger nära blocken så att användaren ska förstå att de hör till just detta block. Knapparna ligger under blocken eftersom vi tycker det känns mest naturligt att man först läser texten och sedan ligger knapparna under det block de tillhör. Vi har också valt att implementera knapparna så att ett nytt block hamnar under aktuellt block om

man väljer att addera ett nytt, vilket naturligtvis betyder att knappen borde ligga under blocket.



7- Block-id:

Varje block har ett id-nummer som står skrivet ovanför blocket för att göra det enklare för administratörer att hålla reda på vilka block som behöver ändras, eller vart det ska läggas till information. Vanliga användare kan referera till detta nummer vid problem av olika typer t.ex. om de behöver hjälp. Detta id-nummer används även för att sortera blocken rätt när man lägger in nya block.

8- Lägg till ett första block:

För att kunna lägga till text eller bild innan allt annat innehåll på sidan så klickar man på denna knapp.

9- Redigera, lägga till och ta bort block:

Dessa tre knappar är verktygen en administratör kan använda sig utav för att förändra, lägga till eller ta bort innehåll på hemsidan. Mer information om dem finns i avsnitt 3.1.1 i denna rapport.

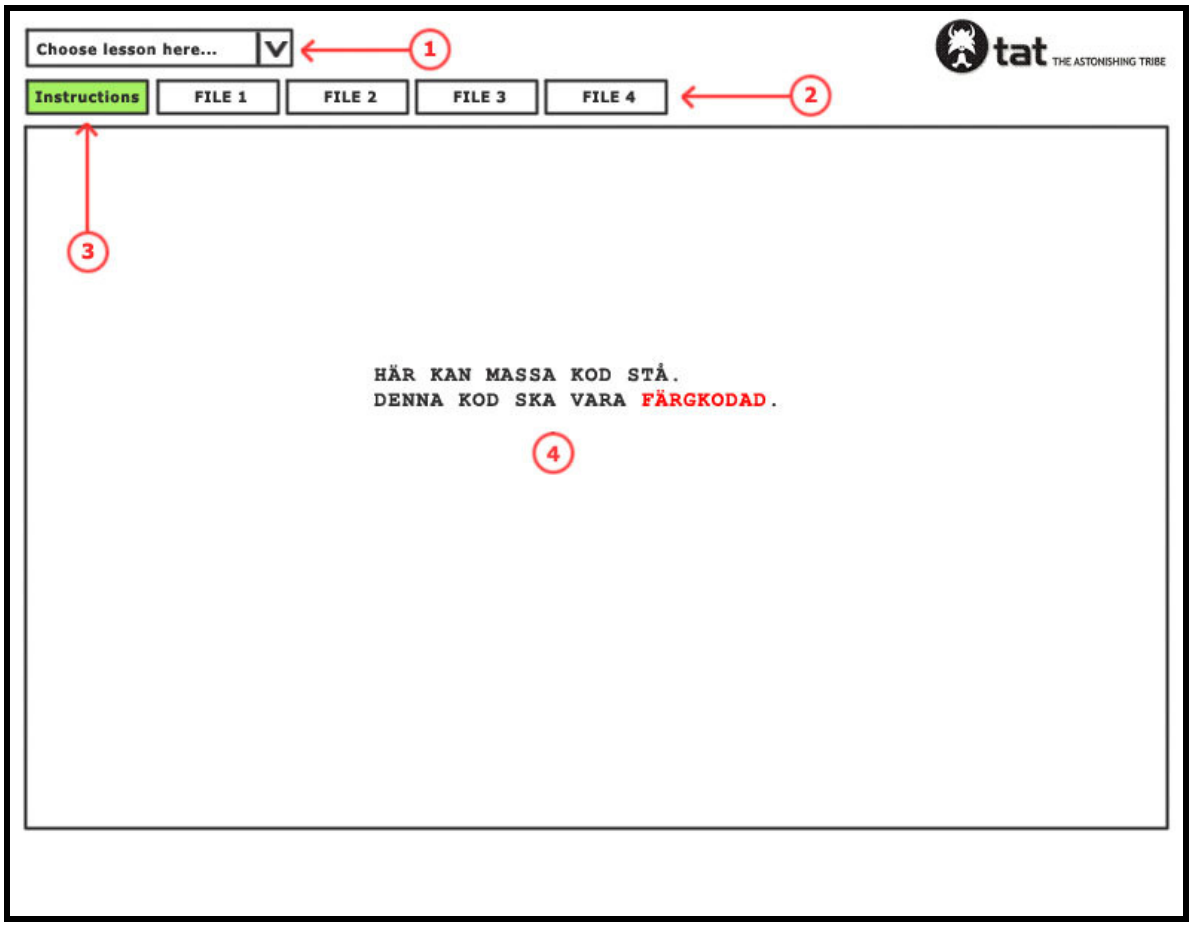
Editorn:

Enligt TATs kravspecifikation så innebär vissa av editorns funktionaliteter ett designmässigt beslut. Dessa beslut har vi tagit utifrån hur vi tycker att applikationen utformar sig bäst. Motiveringar till våra designbeslut kommer vi att ta upp här också.

TATs krav berör huvudsakligen funktionaliteten på editorn och inte dess utseende. Detta har TAT påpekat genom att säga att de kommer att ändra utseendet på editorn om möjligheten finns att den kommer bli användbar för deras kunder. Detta innebär att en del designlösningar kanske inte är de bästa ur ett pedagogiskt perspektiv. Vårt arbete lägger större vikt på det tekniska, funktionsinriktade aspekterna av editorn och tar inte hänsyn till hur vida applikationen blir grafiskt tilltalande eller pedagogiskt väl utformad.

3.2.3 Editorns presentation av övningar

Här nedan är en skiss på hur editorn kommer att se ut. Vi förklarar först de knappar och funktioner som berör visning av övningsfiler och dess innehåll. Varje pil har en siffra som motsvarar den förklaring som står under skissen.



1- Ett av TATs krav är att alla övningar till editorn ska finnas tillgängliga i editorn så att användaren enkelt kan välja en övning de vill köra.
Vår lösning på detta krav är att alla övningar i editorn ska vara tillgängliga i en dropdown-meny som listar dem i nummerordning. Användaren får på sådant sätt en god överblick över vilka övningar som finns och hittar enkelt

den övning hon vill köra. Vi har valt att placera den längst upp till vänster med tanke på att det är där man börjar titta på ett dokument och sedan läser man åt höger och neråt.

2- En övning kommer att bestå utav minst en kodfil och oftast så kommer det att finnas flera. Användaren måste kunna växla mellan de olika kodfilerna för att kunna se deras olika innehåll. Att ha flera textfält skulle kräva en för stor yta och blir svårt att få en bra överblick med så mycket text. Editorn bör därför endast ha ett textfält i vilket de olika filernas kod visas (se punkt 4). För att användaren ska kunna växla mellan de olika filernas kod så har vi bestämt att varje kodfil ska ha en motsvarande knapp som är märkt med kodfilens namn. Klickar man på denna knapp så kommer innehållet i filen att visas upp i textfältet. Dessa knappar dyker upp först när användaren valt vilken övning hon vill köra i dropdown-menyn (Punkt 1). Att ha knapparna precis under dropdown-menyn skapat känslan av en undermeny till övningen vilket dessa knappar i stort sett är.

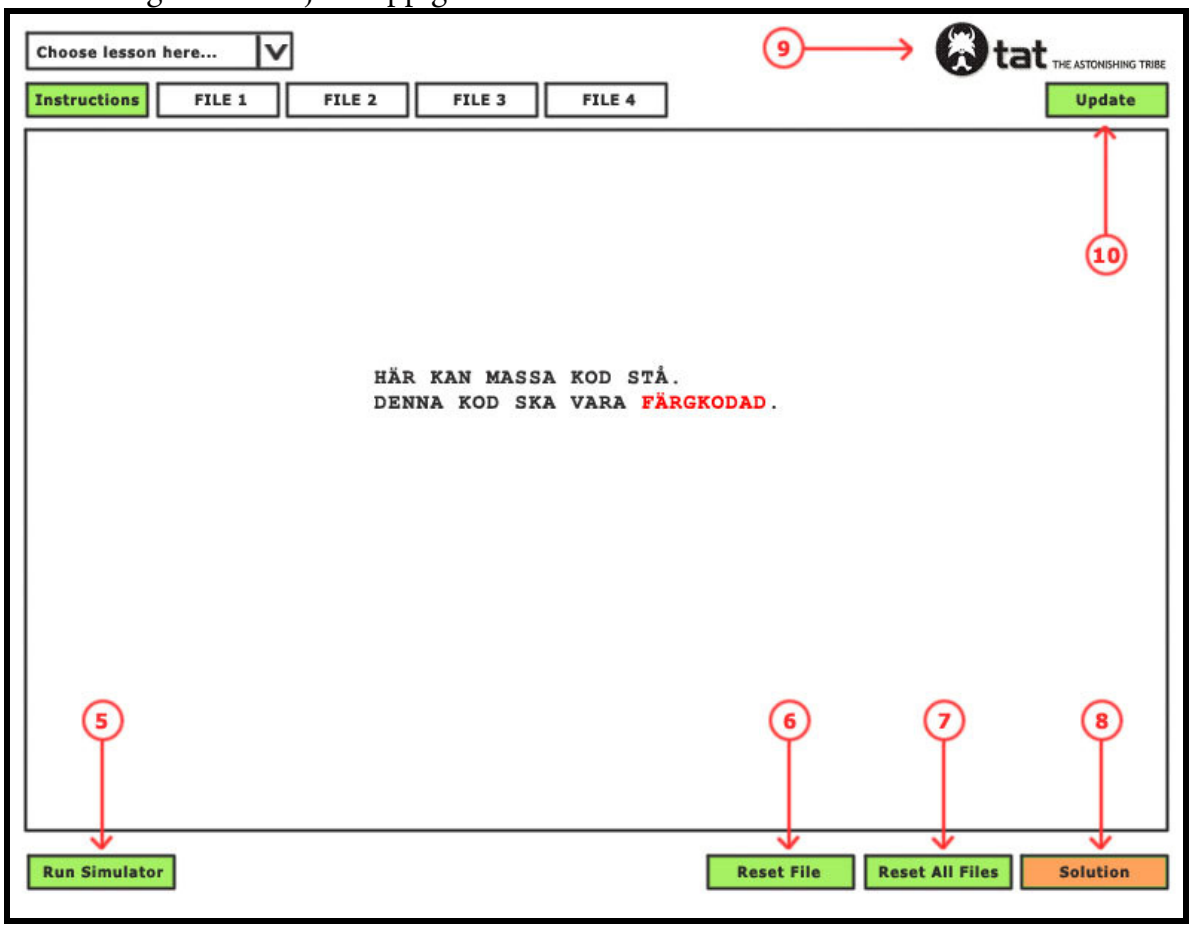
3- På hemsidan så kommer övningarna att ha instruktioner om hur man ska gå tillväga för att lösa dem. TAT har ett krav om att dessa instruktioner ska finnas tillgängliga för användaren i editorn. Detta för att undvika massa hopp fram och tillbaka mellan hemsidan och editorn. Instruktionerna kan då visas i editorns textfält när användaren har valt en övning. Om användaren vill se instruktionerna vid ett senare tillfälle under övningens gång så behövs det en knapp för att återkalla instruktionerna till textfältet.

4- All kod som finns i en övningsfil behöver ett textfält där koden kan visas upp. I textfältet så kommer koden att gå att ändra på. Om man skiftar mellan olika kodfiler så ska innehållet i textfältet sparas undan innan den byts ut. Detta för att användaren ska kunna titta i flera filer utan att förlora det hon ändrat. Detta textfält kan användas för att vissa annan nödvändig information såsom övningens instruktioner. Textfältet kan även användas till att visa eventuella felmeddelanden som kan uppstå. Mer om det i implementeringsavsnittet av denna rapport.

3.2.4 Editorns hantering av kod

Poängen med editorn är att man ska kunna ändra befintlig kod och sedan se resultatet av sina ändringar, samtidigt som man inte ska vara orolig för att kunna förstöra något då man kan återställa all kod till sitt ursprungliga skick med ett enkelt knapptryck.
Alla de knappar som möjliggör detta har vi valt att placera under kodtextfältet för att skilja på deras funktionaliteter med övningens olika fil-knappar. Dessa

knappar är numrerade från 5 till 8 på skissen här under. En mer utförlig beskrivning av vad varje knapp gör hittas under skissen.



5- **Simulatorn:**
Koden ska gå att köras i TATs GUI-simulator med ett knapptryck, de ändringar man gjort i koden ska då visas.

6- **Återställa kod i en fil:**
Koden i en övningsfil ska gå att återställa till ursprungligt läge som när man började övningen.

7- **Återställa koden i övningens alla filer:**
All kod i en övning, det vill säga koden i alla filer som tillhör övningen ska gå att återställa till ursprungligt läge som när man började övningen.

8- **Visa lösning:**
Om man vill veta lösningen på en övning så ska det gå att få fram det med hjälp av en knapp som ändrar innehållet i alla kodfiler till den rätta lösningen.

9- Hjälpfil:

TAT-logotypen ska fungera som en länk till en hjälpfil till editorn. Om en användare är osäker på hur man använder editorn och inte är inloggad på hemsidan så är denna genväg till hjälpfilen en lösning. I hjälpfilen står det kortfattat hur man använder sig utav de olika knapparna.

10- Uppdatera editorn:

Med hjälp av denna knapp kan användaren kolla om hon har den senaste versionen av editorn.

3.3 Programmeringsspråk till hemsida och editor

Programmeringsspråk till hemsidan:

Eftersom vi har bra insikt i vad man kan åstadkomma med olika typer av språk för webbutveckling har vi valt programmeringsspråk att utveckla hemsidan med som vi vet uppfyller de krav som är ställda på hemsidan. Vi beskriver och motiverar de utvecklingsmiljöer som valts här.

Hemsidan kommer att optimeras för Internet Explorer eftersom det är en av de mest använda Internetläsare på marknaden. Enligt internet-sökmotorn Google[14] så har 57% av dess besökare Internet Explorer. Funktionaliteten kommer att testas i olika Internetläsare och ska fungera bra även i dessa.
Databasen görs i SQL med MySQL som databashanterare(DBMS).
Kommunikationen mellan Internetläsare och databas sköts med hjälp ut av PHP.
JavaScript kommer att användasför att skapa allt det interaktiva på hemsidan, dvs de funktioner som möjliggör öppna och stänga menyval i menyn, olika utseendeförändringar då musen förs över något grafiskt element (mouse-over) och kontroll av Internetläsare för att ladda rätt CSS fil.

3.3.1 Hemsida – HTML (HyperText Markup Language)

HTML är ett märkspråk och en webbstandard för strukturering av text, hypertext, media och inbyggda objekt på exempelvis webbsidor och i e-postmeddelanden. Detta språk utgör själva stommen av hemsidan, det skapar skelettet där man sedan skriver in kod från andra programmeringsspråk för att göra hemsidan mer dynamisk och interaktiv.

Referenser:
http://www.w3.org/TR/html401/ 30 juli, 2007
http://www.w3schools.com/html/html_intro.asp 26 nov, 2007

3.3.2 Hemsida – PHP

PHP är ett server-inriktat programmeringsspråk. Detta innebär att php-koden ligger på servern och en användare (klient) kan inte komma åt dessa filer utan endast tolkningen av dem. Detta gör PHP till ett säkert alternativ om man vill dölja information för användaren vilket kan användas till ett inloggninssystem.
PHP anses vara ett av de snabbaste språken för att kommunicera med en databas.
Språket är en öppen källa (Open Source) vilket innebär att inga licenser krävs för att använda det i kommersiellt syfte. Detta gör det enkelt att hitta många lösningar till problem som andra råkat ut för på Internet.
Språket fungerar på de flesta stora plattformer som finns på marknaden, t.ex. Windows och Mac.

[14] Källa: http://www.w3schools.com/browsers/browsers_stats.asp 27 nov 2007

PHP är kod som tolkas på serversidan, vilket innebär att man måste ladda upp koden till en server för att kunna testköra den. Språket har inte heller någon välutvecklad felhantering om man gör fel i koden. Dessa två punkter kan göra utvecklingen av produkten en aning besvärlig.

Referenser:
http://www.devarticles.com/c/a/PHP/PHP-For-Dummies/1/ 23 maj, 2007
http://www.php.net 23 maj, 2007
http://databases.about.com/cs/development/g/php.htm 3 juli, 2007
http://www.webopedia.com/TERM/P/PHP.html 3 juli, 2007

3.3.3 Hemsida – JavaScript

JavaScript är bland de populäraste webbläsar-scripting språken. Det möjliggör både utseendeförändring och ökad funktionalitet på hemsidor. Språket används mest på klientsidan vilket innebär att det körs av Internetläsaren. JavaScript ska inte jämföras på något sätt med programmeringsspråket Java.

JavaScript möjliggör ”riktig” programmering i html-dokument. Detta innebär att man kan deklarera funktioner samt använda programmeringstermer såsom ”if”-satser och ”for”-loopar. JavaScript gör det också möjligt att ha funktioner körandes i bakgrunden i realtid under tiden användare besöker hemsidan. Om man vill göra en visuell förändring efter ett knapptryck så behöver inte sidan laddas om vilket gör navigering smidigare för användaren. En nackdel är att JavaScript tolkas lite annorlunda beroende på webbläsare. JavaScript går även att stänga av på en hemsida, vilket medför att scripten ej körs.

Referenser:
http://www.w3schools.com/js/js_intro.asp 23 maj, 2007
http://www.mediacollege.com/internet/javascript/pros-cons.html 23 maj, 2007

3.3.4 Hemsida – SQL (Structured Query Language)

SQL är ett programmeringsspråk som är designat för att skapa relationer(tabeller) samt att hämta och sköta data från en databashanterare (DBMS) på olika sätt. Databasen kommer att skötas med MySQL som databashanterare. Databasen kommer att användas för att spara innehållet till hemsidans olika handledningstillfällen, login med tillhörande lösenord och en del annat. SQL ska användas för att skapa, uppdatera och ta bort rader med information i de olika relationerna.

Referenser:
http://www.w3schools.com/sql/sql_intro.asp 26 nov, 2007

3.3.5 Hemsida – CSS (Cascading Style Sheets)

CSS är ett programmeringsspråk som används för att bestämma hur innehållet skapat av ett märkspråk, som HTML i detta fall, ska presenteras. Med CSS kommer de olika objekten som ska placeras ut på hemsidan att positioneras. Med detta programmeringsspråk kommer även saker som fonter, muspekarens form, text
dekorering, färger, om bilder ska repeteras eller inte, m.m. att bestämmas.

Referenser:
http://isp.webopedia.com/TERM/C/CSS.html 26 nov, 2007
http://www.w3schools.com/css/css_intro.asp 26 nov, 2007

Programmeringsspråk till editorn:

När vi valde språk att använda för att implementera designsketchen till editorn så gick vi igenom en del olika programmeringsspråk. Ganska snart kom vi fram till att HTML Applikations skulle räcka för att implementera de funktioner som krävs av applikationen utifrån kraven ställda i kravspecifikationen. Eftersom HTA Applikationer använder sig av språk som vi är bekanta med är detta en stor fördel.

Vi listar de plattformar som diskuterats här i tur och ordning. Först kommer en övergripande beskrivning om vad det är för plattform. Därefter beskrivs för och nackdelar. Sist i avsnittet efter att alla olika plattformar presenterats så motiveras valet vi slutligen gjorde mer utförligt.

3.3.6 HTML Applications

HTML Applications (HTA) är applikationer som packat ihop egenskaper från Internet Explorers objektmodell med dess prestanda, presentation, protokoll support och nerladdningsteknik, utan de strikta restriktioner som Internetläsaren har på säkerhet. HTML Applikationer är ett enkelt sätt att göra en applikation som presenteras och beter sig mer som ett program än en hemsida. Detta på grung av den visar endast menyer, ikoner, verktygsfält och titel-information som utvecklaren skapar. Detta kan göra det enklare för utvecklaren att förmedla vad användaren ska fokusera sitt intresse på istället för att ha massa extramenyer från internetläsare och andra externa program som kan vara störande. HTA:s kan skapas med hjälp av HTML eller DHTML som är en dynamisk form av HTML (HTML, CSS och JavaScript). HTML Applications stödjer Javascript vilket medför fördelarna och nackdelarna som nämnts senare angående det språket.

Fördelar:
HTA Applikationer stöder inte bara samma funktioner som en webbsida gör (HTML, CSS och olika scriptspråk) utan har också HTA-specifik funktionalitet. Denna tillagda funktionalitet gestaltar sig i form av kontroll av användargränssnitt, och det som vi framförallt är intresserade av här, åtkomst till

klientens lokala system. Detta innebär att exekveringsfilen till simulatorn går att starta direkt från programmet, vilket inte är möjligt på en hemsida. Vidare har HTA inte samma säkerhetsrestriktioner som hemsidor. Detta innebär att HTA-program kan skriva till filer samt läsa filer lokalt.

Nackdelar:
Filändelsen .hta kan se annorlunda ut för en användare som aldrig sett filändelser av denna typ förut, vilket skulle kunna resultera i viss förvirring när själva editorn skall startas. En annan nackdel med denna plattform är att datorn som editorn skall köras på måste ha Internet Explorer för att fungera.

Referenser:
http://msdn2.microsoft.com/en-us/library/ms536471.aspx 21/5

3.3.7 Macromedia Director

Macromedia Director är ett program för att skapa interaktiva presentationer och applikationer. Med hjälp av detta program är det enkelt att placera ut grafiska element samt att animera dem. Det är i likhet med Flash ett WYSIWYG program, men Director är mer till för att göra applikationer än hemsidor. Directors eget programmeringsspråk kallas Lingo, men även JavaScript går att använda. Dessa gör det möjligt att åstadkomma det mesta man med andra programmeringsspråk kan producera, skillnaden är att man utnyttjar koden för att påverka de element man placerats ut i programmet till skillnad från mer avancerade språk där man skapar de grafiska element i själva koden.

Fördelar:
Lingo tillåter att man skriver till filer lokalt och kan köra igång exekverbara filer. Detta är en fördel eftersom TATs simulator är ett program som editorn ska kunna köra igång då användaren vill se resultatet av sin kod. Macromedia Director gör det enkelt att snabbt sätta upp alla nödvändiga element som knappar och inmatningsfält för att sedan enkelt kunna bestämma vilka funktioner de ska ha.
Färgkodning går att implementera med Lingo eftersom Lingo har en funktion där man kan få tag i ett specifikt ord eller tecken i en text. Detta är ett måste för att färgkodningen ska kunna känna igen vissa kodord eller tecken.

Nackdelar:
Macromedia Director är ett program ägt av företaget Macromedia vilket kräver en licens om ett program ska användas kommersiellt. Enligt Macromedia Director 8.5 User Agreement så behövs det en ”commercial” licens för att kunna få sälja en slutprodukt gjord i Director. Denna kostar ca 1000$. Möjligheten finns att vi kan utveckla produkten på skolan med en ”educational” licens och om TAT senare vill ha applikationen vi skapat, så får de betala den kommersiella licensen.

Olika funktioner som Director inte stödjer i första hand kräver att man köper en licens för varje specifik funktion som kommer att användas. Ett exempel på detta är stöd för den mellersta musknappen som oftast används för att scrolla ner och upp på textsidor. Denna funktion är ett måste om man ska göra kodningen lätt att ta sig igenom.

Referenser:
http://www.director-xtras.com/xtra-mouse_control.html 21/5
http://www.lingoworkshop.com/Articles/history.php 30/6
http://www.adobe.com/products/eula/tools/director_mx_2004.html 30/6

3.3.8 Java

Java är ett objektorienterat programmeringsspråk utvecklat av Sun som används både till klientbaserade och serverbaserade applikationer.

Fördelar:
Plattformsoberoende, vilket betyder att applikationer programmerade i Java funkar lika bra på Mac som PC eller i olika internetläsare.
Java sköter minneshanteringen automatiskt, så att programmeraren inte behöver oroa sig för deklarerade variabler som aldrig tas bort när de inte används.
Java är gratis, vilket gör att ingen licens krävs för att få skapa ett kommersiellt program.

Nackdelar:
Java kräver att en egen plattform kallad Java Virtual Machine (JVM) är installerad för att kunna köra Java-program.
Java är lite mer processor-krävande än andra programmeringsspråk (ej skriptspråk som Javascript), bland annat på grund av den automatiska minneshanteringen. Detta gör det inte så lämpligt för program som kräver realtidsuppdateringar, som färgkodning av kod till exempel.
Objektorienterad programmering är väldigt bra när man ska skapa stora projekt, men för simplare program så kan det bli lite omständigt och svårare att få en enkel lösning. Microsoft har slutat samarbeta med Java, eftersom de håller på att utveckla ”Java-dödaren” C-Sharp (C#) som de tycker ska ta dess plats i deras Windows-miljö. Detta kan skapa problem i framtiden om Editorn är programmerad i Java och nyare Windows inte stödjer det.

Referenser:
http://www.pcmag.com/encyclopedia_term/0,2542,t=Java&i=45557,00.asp 3/7
http://www.abrandao.com/cittone.abrandao.com/ref/java2wrap-up.htm 31/7

3.3.9 JavaScript (Editor)

JavaScript är ett populärt förenklat programmeringsspråk som oftast integreras med annan webbdesignkod som HTML. Språket möjliggör realtidsfunktioner som kan köras i bakgrunden av en hemsida som ett riktigt programmeringsspråk till skillnad från HTML som används för att forma det grafiska som ska presentera informationen på en hemsida. JavaScript burkar användas till att förändra utseendet av en hemsidas olika grafiska element utan att hemsidan behöver laddas om då användaren interagerar med den.
I kombination med Html Applications så kan JavaScript få åtkomst till filer och mappar på användarens dator. Detta är nödvändigt om editorns ska köras lokalt och det ska finnas möjlighet att spara och läsa kodfiler.

Fördelar:
JavaScript är ett händelse drivet programmeringspråk, vilket gör det lämpligt för interaktiva program eftersom språket kan reagera på användarens handlingar såsom knapptryckningar utan att behöva ladda om applikationen.

JavaScript exekverar kod på klientsidan vilket har möjliggjort elakartade program på Internet som utnyttjar användarens dator. Detta har fått som konsekvens att användare har möjligheten att stänga av JavaScript i Internetläsarens egenskaper. Det skulle vara förödande för funktionaliteten av editorn om scripten slutade fungera även i HTA dokumentet. Men så är inte fallet eftersom HTA kör skript ändå trots att de stängts av i Internetläsaren.

Nackdelar:
JavaScript exekverar kod på klientsidan vilket har möjliggjort elakartade program på Internet som utnyttjar användarens dator. Detta har fått som konsekvens att användare har möjligheten att stänga av JavaScript i Internetläsarens egenskaper. Det skulle vara förödande för funktionaliteten av editorn om scripten slutade fungera även i HTA dokumentet då script möjligheter stängs av i Internetläsaren. Men även om script stängs av i Internetläsaren så fungerar de i editorn.

Referenser:
http://www.w3schools.com/js/js_intro.asp 23 maj, 2007
http://www.mediacollege.com/internet/javascript/pros-cons.html 23 maj, 2007
http://www.pcmag.com/encyclopedia_term/0,2542,t=JavaScript&i=45585,00.asp 3/7

3.3.10 C++

C++ är ett högnivå-programmeringsspråk som liknar sin föregångare C men stöder till skillnad från denna, objektorienterad programmering. Språket används flitigt av kommersiella företag för att skapa större grafiska omfattande applikationer i Windows och Mac-miljö.

C++ används mest för att bygga vidare på färdiga bibliotek som antingen Windows
erbjuder eller kraftigare grafiska paket som DirectX.

Fördelar:
C++ är ett kraftfullt programmeringsspråk som möjliggör det mesta om man bemästrar språket. En stor arsenal med bibliotek finns tillgängliga för användning på Internet vilket gör det enklare att kunna utveckla mer avancerade program. C++ är plattformsoberoende.

Nackdelar:
En mycket hög inlärningströskel gör C++ tidskrävande att lära sig tillräckligt för att kunna åstadkomma önskvärda program. Man måste hålla reda på minnesallokering/avallokering själv, detta kan leda till att man glömmer att avallokera och får på så vis minnesläckor.

Referenser:
http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci211850,00.html 5/7
http://www.webopedia.com/TERM/C/C_plus_plus.html 5/7
http://www.programmersheaven.com/2/Beginners-Guide-To-C-Plus-Plus 5/7

3.3.11 .NET
.Net ramverket är Microsofts svar på Javasamfundets motsvarighet ”Windonization” av Java. .NET ramverkets CLR (Common Language Runtime) motsvaras av Javas JVM, och har som främsta uppgift att översätta bytekod till maskinkod (Just in Time Compilation).

.NET är i grund och botten en plattform för snabbutveckling av applikationer för både webb och desktop. Det är ett Microsoft-utvecklat ramverk i vilket C#.NET, VB.NET och JScript.NET kan kombineras tack vare deras gemensamma tolkare, CLR. Med .NET får man tillgång till ett stort bibliotek med hjälpklasser som underlättar ytterligare vid utvecklingen.

Microsofts serverorienterade originalprogrammeringsspråk ASP (Active Server Page) anses vara kraftfullt men kräver har en aning högre inlärningströskel än PHP. ASP är ett fristående språk och den senaste versionen av det är 3.0. Genom att skapa ASP.NET har Microsoft försökt göra det enklare och snabbare för användaren att skapa webbapplikationer. ASP.NET är en evolution av ASP byggd runt .NET-ramverket.

Genom att göra det lättare för användaren att programmera så minskar mängden kod och hela arbetet blir mer effektivt och produktivt. .NET separerar design, skript och ”serverside”-programmering och kan på så sätt ge ett utvecklarteam

bättre möjligheter att jobba parallellt. Det är dessutom möjligt att uppdatera webbapplikationer samtidigt som webbservern exekverar den. Detta möjliggörs genom Microsofts teknologi Shadow Copy.

Fördelar:
Stödjer många programspråk däribland, C, C++, JavaScript och Visual Basics.

Nackdelar:
En hög inlärningströskel gör .NET tidskrävande att lära sig tillräckligt för att kunna åstadkomma önskvärda program.
.NET fungerar endast på Microsoft servrar.

Referenser:
http://www.csharphelp.com/archives/archive10.html 31/7 - 2007
http://www.odetocode.com/Articles/305.aspx 31/7 – 2007
http://msdn2.microsoft.com/en-us/library/4w3ex9c2(vs.71).aspx 31/7 - 2007

3.3.12 Webbsida
I detta avsnitt har vi kollat på fördelarna med att bygga in editorn på hemsidan.

Fördelar:
Det finns klara färgkodnings-editorer som fungerar i HTML t.ex. Codepress som finns att ladda ner gratis på http://sourceforge.net/projects/codepress (26nov 2007) . Allt kan ligga på server sidan och är därför väldigt enkelt att uppdatera. Vi kan HTML och de språk som skulle behöva användas om denna lösning valdes.

Nackdelar:
Säkerhetsspärrar förhindrar exekvering av program (t.ex.: exe-filer) på klientdatorn i samtliga Internetläsare, vilket medför att detta alternativ är uteslutet trots dess många fördelar eftersom TATs simulator är ett program som ska användas av editorn.

Slutgiltigt val av utvecklingsmiljö:
Eftersom det står beskrivit i kravspecifikationen att editorn ska kunna köra den skapade koden i simulatorn som är en exe-fil, så faller alternativet om att bygga in editorn på en hemsida med en gång.

Av de alternativ som är kvar är enligt oss **HTML applikationer** att föredra.

I editorer till små/medelstora textmassor räcker prestandan i JavaScript alldeles utmärkt till, dessutom slipper man att ta hänsyn till minneshanteringen vilket man inte hade kunnat göra i C++. Java är plattformsoberoende, men måste ha sin

plattform JVM installerad för att kunna exekvera koden. HTML Applikationer måste ha Internet Explorer för att fungera, men eftersom simulatorn bara fungerar på Windows datorer tycker vi inte detta är ett allvarligt problem eftersom de allra flesta Windows maskiner har Internet Explorer.

HTML Applikationer gör det också enkelt att skapa själva editorn och dess ramverk. Även om det finns ramverk att använda till även kompilerande programmeringsspråk som c++ eller java, så är dessa betydligt besvärligare att använda. En annan fördel med HTA Applikationer är att vi sedan innan kan DHTML.

JavaScript kan lösa alla design- och funktionella krav vi har på editorn och eftersom vi redan är bekanta med JavaScript så undviker vi en hög inlärningströskel som de andra språken har, då vi kan dem mycket sämre eller inte alls. Dessutom är JavaScript ett väldigt populärt språk och tillgängligheten av information är stor vilket gör det enklare att utöka våra kunskaper.

3.4 Skapande av testmiljö

Varken filer eller databas kommer att ligga på målservern, dvs. TAT's server. Detta eftersom vi inte vet om TAT ska använda applikationen och för att minimera risken för att förstöra något på deras server såsom tabeller från deras andra databaser eller andra fel som kanske skulle kunna uppstå. På grund av detta så har en så kallad testmiljö upprättats på våran egen server.

Hemsidan:

Det står i kravspecifikationen beskrivet att hemsidan ska bestå av information som dynamiskt ska gå att förändra. Det står även att man ska ha kunna lägga till ny information och ta bort gammal. Vi har valt att lösa detta genom att använda databaser där informationen sparas undan. MySQL används som databashanterare och för att komma åt data i relationsdatabaserna via Internet så används programmeringsspråket PHP.

Databasfunktionalitet:

Texten som sparas undan är som tidigare förklarats innehållet i de olika handledningstillfällen som ges. Tillsammans med denna text finns kolumner som förklarar till vilket handledningstillfälle (lektion) ett block tillhör, vilket kapitel inom lektionen och vilken plats respektive block har(1,2,3 etc.). Tillsammans bildar dessa tre siffror ett unikt block-id ex: 1.3.4 som är block 4 i kapitel 3 i lektion 1.

Här finns även tre andra kolumner. En som håller reda på vem som senast förändrade en text, en som kontrollerar om ett block editeras just nu och skall vara i "editmode" för användaren samt en sista kolumn som håller reda på vilken typ av block det är (text, bild eller kodblock) som förklarades i 3.1.1 dynamiskt innehåll.

Informationen som sparats undan i databasen hämtas sedan för antingen visning på hemsidan, eller editering. Editeras texten så hämtas den nuvarande från databasen, därefter finns möjligheten att förändra den tills man är nöjd, varvid den nya texten ersätter den gamla i databasen. ID numret för blocket håller reda på platsen ett block ska ha och därför kan man lägga till ett block på vilken plats man vill i ett kapitel.

3.5 Implementering

Hemsidan:

Under projekttiden har det producerats en färdig-utvecklad prototyp av webbplatsen. Denna finns tillgänglig på:
http://www.jonathan-stahl.se/1EXJOBB/login.php

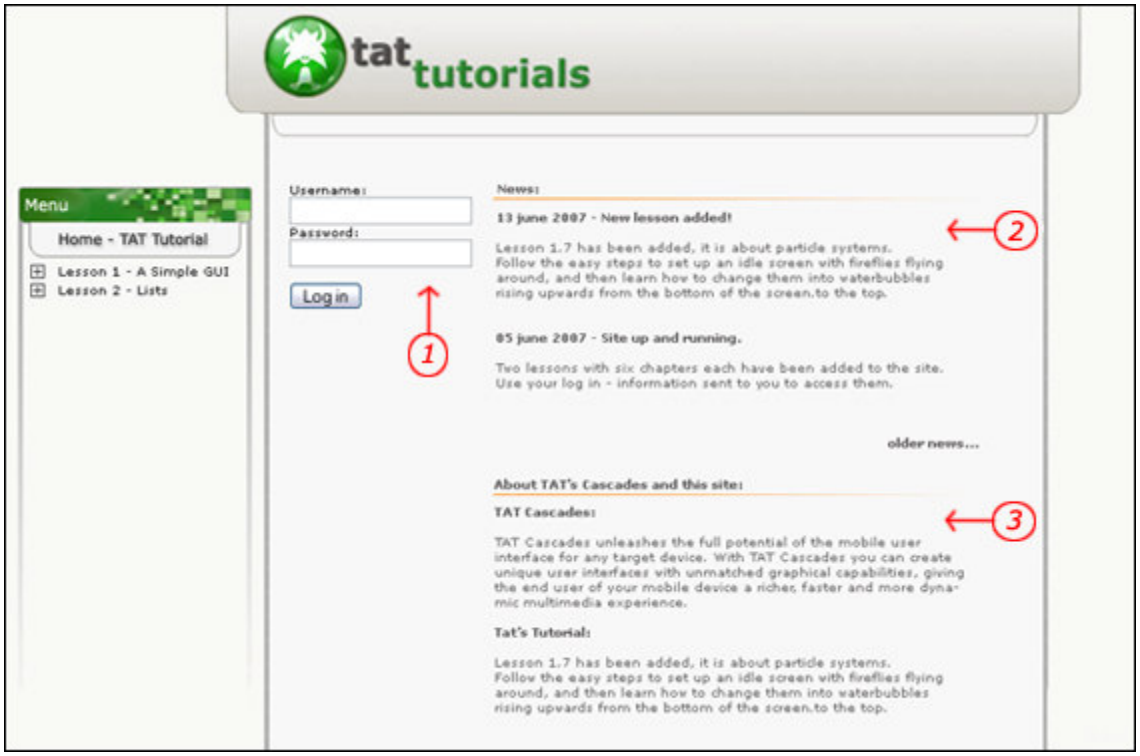
3.5.1 Grafiskt gränssnitt, hemsida:

För att användarna ska ha möjligheten att interagera med hemsidan så behövs ett gränssnitt. Layouten till detta gränssnitt är skapad enligt sketchen som gjordes under kapitel 3.2 om design.
De olika grafiska objekt som finns tillgängliga är alla skapade i Adobe Photoshop och sedan utplacerade i Internetläsaren med hjälp av html. CSS har använts för att positionera objekten.

3.5.2 Teknisk utveckling, hemsida:

Hemsidan består i princip av två huvudfiler, sidor för login och index. Dessa två filer använder sig i sin tur av en del andra filer. Här nedan visas bilder på hur den färdiga hemsidan ser ut samt hur dess olika funktioner och filer är uppbyggda. Alla kodfiler finns i sin helhet i appendix C.

Loginsidan (Login.php):



Denna sida är en login-sida. All kod skapas med html som har inbäddade PHP funktioner för databaskoppling och kontroll av olika variabler.

1- **Inloggningsfält:**
När en användare skriver in rätt användarnamn och lösenord så skickas hon vidare till index-sidan. I övriga fall ges ett fel meddelande om att användarnamn eller lösenordet var fel och man skickas inte vidare.

Login-filen använder sig även av en annan fil som inkluderas. Denna fil har en funktion som behövs ifall magic_quotes_gpc inte är satt. Magic_quotes_gpc (gpc = Get/Post/Cookie) är en PHP funktion som när den är på, sätter ett \ (backslash) framför alla ” (citat tecken), ’ (enkel-citat tecken), \ (backslash) och NULL’s. När ett \ (backslash) sätts framför ett tecken på detta sätt i PHP betyder det att PHP ska strunta i att tolka detta tecken som kod. Detta görs för att förhindra en säkerhetsrisk som uppstår i databaslagret av en applikation. Säkerhetsrisken uppstår när användarinmatning antingen är fel-filtrerad från specialtecken som används för att hämta information i databasen ex: ’(enkel citat) och ”(citat tecken) eller att användarinmatningen är svagt typad. En variabel är svagt typad om den kan ha värden från olika datatyper. Variabeln $TAT är svagt typad om den både kan ha värdet ”55” och ”The Astonishing Tribe”, dvs den kan vara både en siffra och en sträng. Detta kan leda till att något oväntat exekveras i databasen. En elak användare kan alltså skicka in olämpliga SQL kommandon till databasen och på så sätt få reda på känslig information, detta kallas för SQL-Injections.

I lindrigare fall kan funktionen i filen som inkluderas i login.php förhindra MySQL fel.

2- **Nyheter:**
Här finns utrymme för ett nyhetssystem. Nyhetssystemet går inte att uppdatera via hemsidan eftersom man inte är inloggad. Nyhetssystemet kan vara bra då man vill uppmärksamma användare om ändringar, nyheter och annat som kan vara av intresse för den vanlige användaren. Här är ingen funktion implementerad utan detta är bara en bild.

3- **Information om hemsidan och TATs Cascades:**
Här visas information och instruktioner för förstagångsanvändaren om TAT Kastor och Cascades samt TAT Tutorials. Här är ingen funktion implementerad utan detta är bara en bild.

Huvudsidan (Index.php):
Index-sidan är en dynamisk sida som ritas upp beroende på vad användaren gjort för val. Den länkar inte vidare till andra filer, utan allt visas på index sidan. Informationen som visas hämtas från en databas. För att lättare förklara hur de olika funktionerna implementerats tar jag sketchen skapad i designkapitlet 3.2 till hjälp.

Det finns två olika typer av konto man kan logga in med. De som kan lägga till och förändra innehåll samt de som inte kan det. Först beskrivs funktionerna i ett vanligt användarkonto.

Huvudsidan - Användarkonto:

4- **Menyn:**
Menyn är en trädlista som kan fällas ut och in så att den inte tar så stor plats. Den består av olika handledningstillfällen som i sin tur består av olika kapitel.
Vid knapptryckning på en länk i menyn så laddas index-sidan om, PHP variabler med värden som korresponderande till det handledningstillfälle och kapitel man tryckt på skickas med. Informationen som skall presenteras hämtas sedan från en databas och visas. Ut- och infällning av menyn sköts med hjälp av ett JavaScript som hämtats från: http://www.dynamicdrive.com/dynamicindex1/navigate1.htm 070617

5- **Inloggningsinformation:**

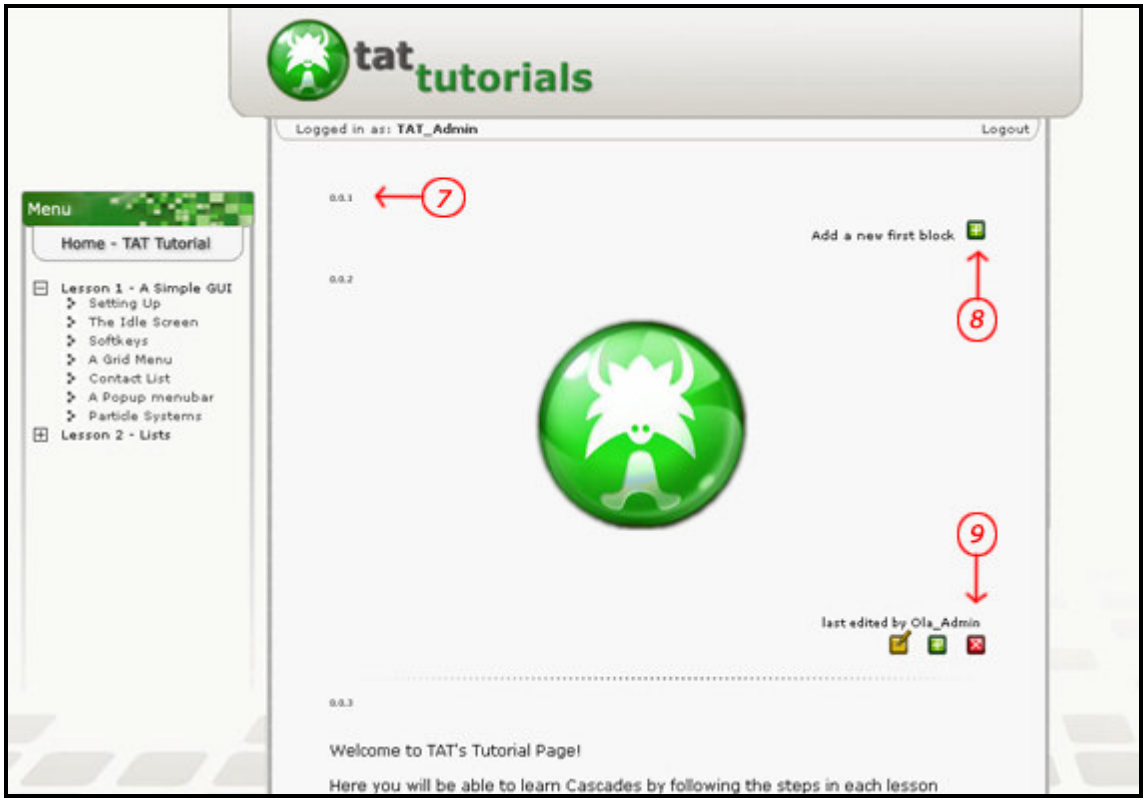
I informationsfältetet finns möjlighet att logga ut samt att se vem man är inloggad som. Vem man är inloggad som finns sparat i en sessionsvariabel(skapad vid inloggning) och skrivs ut på vänster sida . Vid utloggning skickas man till login-sidans logout funktion som förstör alla sessionsvariabler och sessionen själv.

Vid laddning av index-sidan får sessionsvariabeln time limit ett värde. Detta eftersom hemsidan automatiskt laddas om med jämna tidsintervall och om användaren varit borta från datorn tillräckligt länge så blir hon automatiskt utloggad. Alla dessa tre funktioner styrs med hjälp av PHP.

6- **Brödtext:**

Brödtexten representeras som beskrivits tidigare av tre olika typer av block. Vilken typ ett block är av, bestämmas vid undansparningen av blocket. Hur man sparar block av olika typer beskrivs under admin-konto. Hur informationen skall presenteras finns i databasen och all information hämtas
därifrån.

Huvudsidan - Admin-konto:



Admin-konton fungerar precis som vanliga användarkonton förutom att dom har extra funktioner för att lägga till, ta bort och editera block. En if-sats kontrollerar om användaren är admin och om så är fallet så ritas ikoner för de olika funktionerna ut. Om en användare är admin eller ej står beskrivit i databasen.

7- **Block-id:**
Innan varje block presenteras dess block ID. Detta hämtas ur en kolumn i en tabell som heter content, och har sidans alla handledningstillfälle som innehåll. Där finns en kolumn för att veta vilken lesson blocket tillhör, en för kapitel och en för vilken plats blocket har i kapitlet. När blocket sparas undan så läggs dessa siffror ihop till ett block ID och sparas undan i ytterligare en kolumn. Det är denna siffra som sedan hämtas och presenteras här.

8- **Nytt förstablock:**
Vid knapptryckning på addering av nya block blir man frågad om vilken typ av block man vill lägga till. Detta eftersom det beroende på vilken typ av block man valt att addera, läggs in taggar innan och efter texten så att blocket senare skall visas i sin rätta form. Innan textblock läggs inga taggar till.

Till kodblock läggs följande kod in innan innehållet i blocket:

```
<textarea name=[blockID x.x.x] cols=63 rows=20
wrap="off" id="cp" class="codepress html
linenumbers-on">
```

Och följande kod läggs till efter:

```
</textarea>
```

Det viktigaste att veta här är att det som står efter `class` bestämmer hur innehållet i codepress-textrutan ska visas. Codepress är JavaScriptet som färgkodar och sätter ut radnummer i kanten, vilket gör det enklare för användare att behandla kod, då dess olika element blir tydligare och radnummer underlättar sökandet i dokumentet.

Bild blocken lägger in detta före innehållet:

```
<img src="img/
```

Och detta efter:

```
"></img>
```

Popupen som visas vid knapptryckning på addBlock knappen styrs med hjälp av ett JavaScript.

I länken till addBlock.php skickas variabler med som beskriver vilket block det är man vill lägga till ett nytt efter. I detta fall alltid 1 eftersom det är det första blocket. När ett nytt block läggs till är det viktigt att det nya block-id:t sorteras in rätt. Block-id är den siffra som presenteras i punkt nummer 7 i sketchen och det är med denna siffra som sidan räknar ut i vilken ordning den ska presentera blocken. Block-id:t är uppbyggt av numret på handledningstillfället, kapitlet och blockets plats i kapitlet, till en unik siffra för varje block. Det betyder att 1.2.3 är block 3 i kapitel 2 i handledningstillfälle 1. När ett nytt block läggs till som 1:a block så ökar bara den tredje siffran i alla andra blocks id:n med 1. Efter detta läggs det nya block till som x.x.2.

När ett nytt block av en viss typ lagts till sätts det genast i editerbart tillstånd så att man kan skriva innehåll i det. Hur editerbart tillstånd hanteras förklaras under ”Edit” avsnittet i punkt 9.

Grunderna till popup-JavaScriptet som används har hämtats från:

http://developer.apple.com/internet/webcontent/hideshow_layer.html
070617

Rad 208-237 i Index.php berör addering av nytt förstablock.
Addering av nya block använder sig utav filen addBlock.php.

9- Add, Edit och Delete:

Under varje block finns tre små knappar. En för att lägga till ett block efter det förra, en för att editera text och en för att ta bort block. Att lägga till block fungerar precis som att lägga till ett nytt förstablock med undantag av att endast blocknummer som kommer efter det aktuella blocket ökas med 1.

9- Edit:

Om man trycker på den gula edit-ikonen skickas variabler till editBlock.php som berättar vilket block det är man vill editera. Editblock filen uppdaterar en kolumn i databasen som håller reda på om ett block är i editerbart tillstånd eller inte till ”1”. Att den får siffran ”1” betyder att texten just nu editeras. Siffran ”0” betyder att det inte editeras. I indexfilen finns där en variabel som sedan hämtar detta värde och kontrollerar om något block är i editerbart tillstånd, är fallet så och användaren har admin-rättigheter så ritas en form upp. Formen får blockets innehåll som editerbar text. Till detta tillstånd kan man komma på två sätt, antingen genom att ha tryckt på edit-knappen eller efter att ha lagt till ett block. Båda hamnar i editerbart tillstånd.

Under formen finns två knappar. En som heter ”accept” och en som heter ”cancel”. ”Accept”-knappen leder till en fil som heter editDone.php.
Om man kommer från addering av block när denna knapp trycks ner så läggs de typ-specifika taggarna till innan och efter innehållet, sedan sparas innehållet undan i databasen. Kommer man däremot till editerbart tillstånd efter att ha tryckt på edit-ikonen så läggs inga extra taggar till, eftersom dom då redan finns där. Ett undantag är dock kodblock, där man behöver stänga textarea-taggen igen även om man kommer från edit.

Väljer användaren istället att trycka på ”cancel” så sparas inget undan utan sidan ändrar bara blockets status till icke editerbart och går tillbaks till visningsläge. Är fältet tomt betyder det att man kom från addering av block och man skickas nu till removeBlock.php som tar bort blocket. Mer om ”remove block” under dess kapitel här nedan.

9- Remove:

Vid knapptryckning på den röda remove ikonen så tillfrågas användaren först av en popup om hon verkligen vill ta bort detta block. Trycks

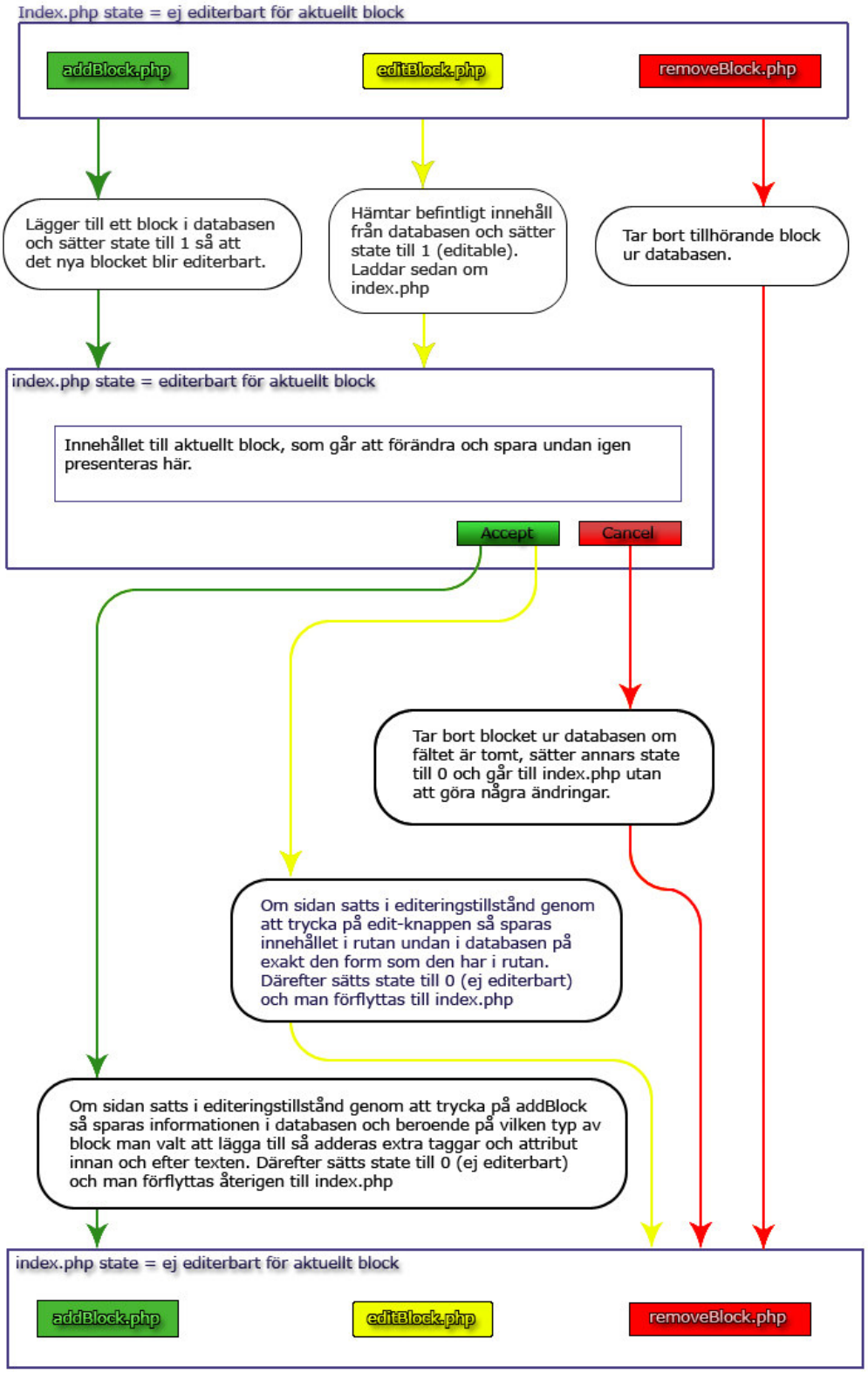
knappen ”yes” ner så skickas hon till removeBlock.php. I URL:en skickas då även id-numret på blocket borttagningen gäller.

I removeBlock.php raderas aktuellt block från databasen och efterföljande block får sitt blocknummer reducerat med 1. På grund av detta förändras även dess block-id, var på användaren skickas tillbaks till index-sidan där hon innan var.

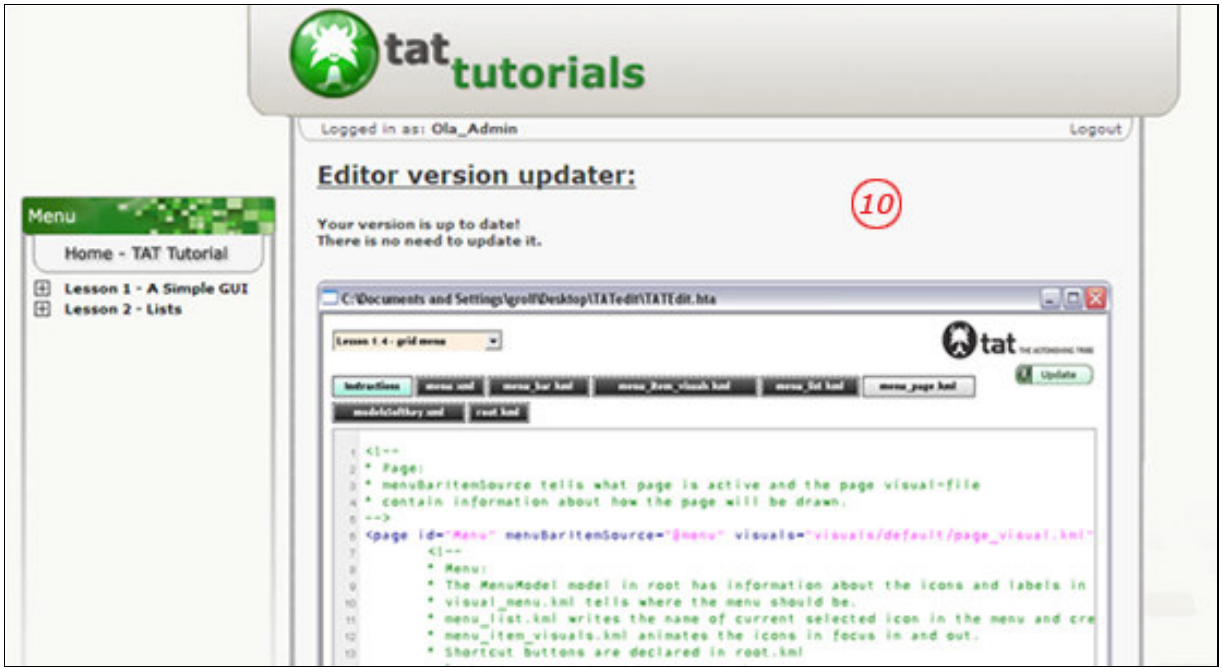
Grunderna till popup JavaScriptet som används har hämtats från:
http://developer.apple.com/internet/webcontent/hideshow_layer.html
(7 juni 2007)

Här nedan är ett flödesschema över hur funktionerna för addering, editering och borttagning av block fungerar:

3.5.3 Flödesschema av blockfunktionerna på hemsidan:



Uppdatering av editorn (update.php):



10- Editor uppdatering:

När man i editorn trycker på uppdateraknappen så transporteras man till en extern länk som ligger på hemsidan. När knappen trycktes ner i editorn så skickades variabler med som berättar vilken editor version användaren har samt att man kom till denna sida genom att trycka på updateknappen. Att skicka variabler i URL:en på detta vis är naturligtvis inte säkert nog och därför måste användaren även vara inloggad för att uppdateringen skall vara möjlig.
Skulle en användare trycka på update i editorn utan att vara inloggad så kommer hon till inloggninsskärmen och en text berättar att man behöver vara inloggad för att ha möjlighet att uppdatera editorn. Eftersom editorn är gjord som en HTML applikation, vilket är en produkt av Microsoft så öppnas updatelänken i Internet Explorer. Detta innebär att man även måste vara inloggad och ha en session i Internet Explorer. Uppatera-funktionen fungerar alltså inte i andra Internetläsare.

Uppdateringsfunktionen laddas i update.php genom en switch sats som startar olika funktioner beroende på vilken version av editorn man har. Funktionen som sedan startas kollar mot en databas vilken version som är den senaste. Har man en version som är gammal ges en länk där den nyaste editorn går att ladda hem. Om man har en editor som inte behöver uppdateras så meddelas användaren även om detta.

Editorn:
Editorn finns tillgänglig för nerladdning på vår prototyp-sida:
http://www.jonathan-stahl.se/1EXJOBB/TATedit.zip (20 juli 2007)
Man startar editorn genom att packa upp zip-filen med WinZip eller Microsofts Windows inbyggda zip-hanterare, och sedan starta filen ”TATedit.hta”.

3.5.4 Grafiskt gränssnitt, editor:
Alla grafiska element som knappar, dropdown-meny och inmatningsfält är html-form element förutom bakgrunds bilden som är skapad i photoshop. Elementens färger och placering är ordnade med CSS enligt sketchen som gjordes under design-kapitlet 3.2.

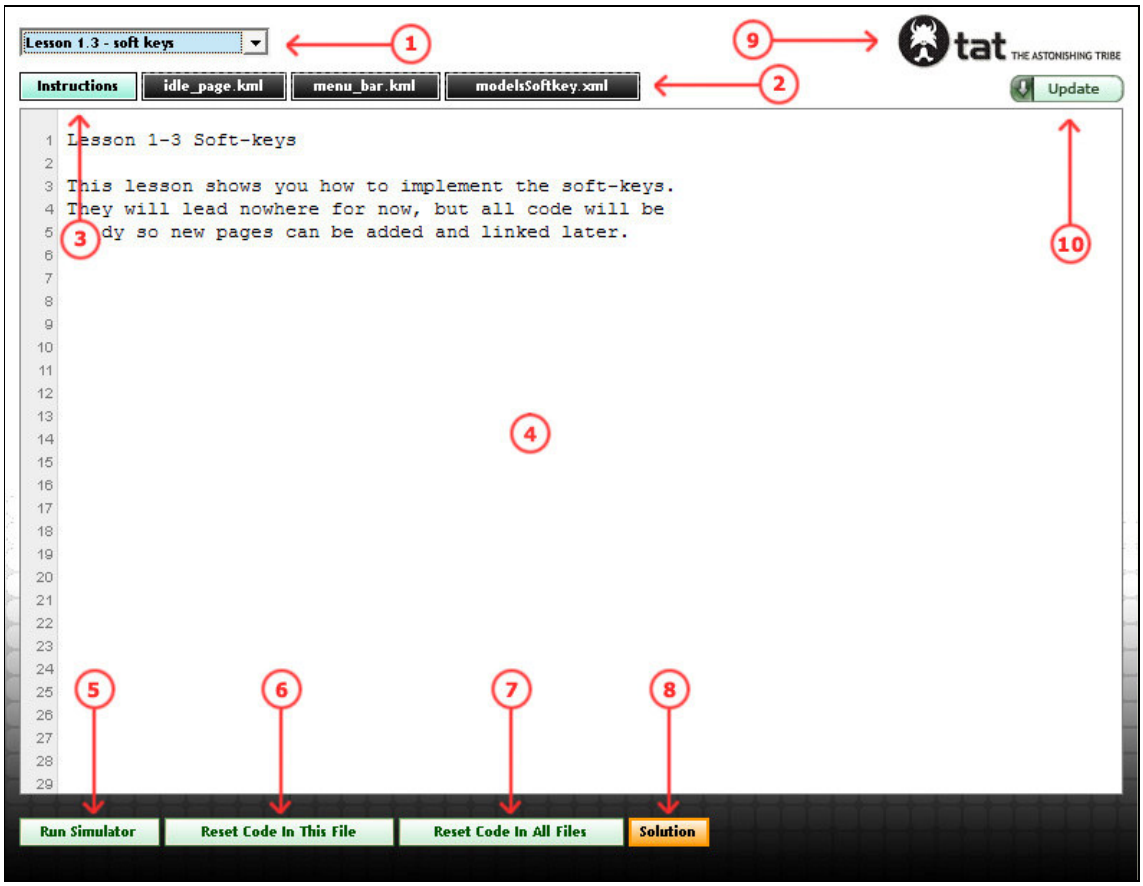
3.5.5 Katalogstruktur, editor:
Innan alla editorns funktionalitet tas upp och hur de blivit implementerade så är det viktigt att förstå hur editorns katalogstruktur är skapad och hur de olika filerna kommunicerar.
Efter att ha packat upp editorn till en katalog på hårddisken (ex: ”C:\TATedit\”) så ligger startfilen TATedit.hta direkt i denna mapp (”C:\TATedit\TATedit.hta”).

I samma katalog ligger TAT’s simulator som heter CascadesViewer.exe. Detta program används för att köra igång kod man skrivit i editorn för att få ett grafiskt resultat på hur ett GUI på mobiltelefonen skulle se ut.
För att CascadesViewer.exe ska fungera så måste den ha en inställnings-fil liggandes i samma katalog som sig själv. Denna fil heter ”settings.xml” och finns i editorns rotkatalog tillsammans med TATedit.hta och CascadesViewer.exe (”C:\TATedit\settings.xml”) .
Denna settings-fil innehåller alla sökvägar till de filer som bygger upp ett GUI och som simulatorn behöver för att kunna köras.

Alla övningar som går att göra i editorn ligger i katalogen Lessons som ligger direkt i editorns rotkatalog (”C:\TATedit\Lessons\”). Varje övning är en underkatalog som ligger i denna Lessons-mapp. I varje övningskatalog ligger en settings-fil. Denna settings-fil innehåller sökvägar till de filer som simulatorn ska använda i denna övning.
Simulatorn CascadesViewer.exe läser endast in settings-filen som ligger i samma katalog som den själv. För att simulatorn ska kunna få informationen som ligger i de andra settings-filerna som är i varje övningsmapp så kopieras innehållet i en övnings settings-fil in till settings-filen som ligger i rotkaltalogen med simulatorn. Detta sker när användaren väljer övning i editorn, mer om det i nästa avsnitt om teknisk utveckling.

3.5.6 Teknisk utveckling, editor:

För att göra det enklare att hänga med i vilka funktionaliteter som editorn erbjuder och hur dessa aktiveras så finns här nedan en bild på hur editorn ser ut och alla dess knappar. Varje funktion har ett nummer, numren är listade under bilden tillsammans med förklaringar om hur respektive funktion implementerats.



Här nedan är en lista över editorns funktioner samt en kort beskrivning om hur deras implementation löstes. Kodfilen i dess helhet är listad i Appendix C. Siffrorna i listan representerar siffrorna på bilden ovanför:

1- **Övningslista:**
Lista alla tillgängliga övningar (se funktionen *listLessons() i Appendix C).* Med hjälp av HTML forms så skapas en dropdown-meny. Alla alternativ i dropdown-menyn genereras av en JavaScript-funktion som kollar hur många kataloger som finns i Lessons-mappen i editorns huvudkatalog. Namnen på katalogerna listas i dropdown-menyn.
Exempel:
Två kataloger finns i editorns Lessons-mapp med följande sökvägar:
C:\TATedit\Lessons\TutorialOne
C:\TATedit\Lessons\TutorialTwo
Editorn kommer nu lista följande alternativ i dropdown-menyn:

TutorialOne
TutorialTwo

2- Visning snabbknappar:

Då en övning valts så ska alla dess övningsspecifika filer, dvs. dem i ”show” mappen, visas upp som knappar (Se funktionen *listFiles() i A3).*
När man startar editorn så finns det 10 dolda knappar som senare visas upp beroende på antalet filer i en övningskatalog. För att få knapparna att dyka upp så ändras CSS-koden som styr hur de ska visas (*style='display:none;'* blir *style='display:inline;'*). Knapparna får även namn skrivna i sig som motsvarar filernas namn. Dessa ändringar sköts av en JavaScript-funktion som anropas då man gjort ett val i listan av övningar i dropdown-menyn.

2- Snabbknappar:

Ett tryck på en fil-knapp ska visa innehållet i filen med samma namn som knappen (se funktionen *readFile(filnamn)*).
Då man trycker på en av de knappar som dykt upp när en övning valts så visas innehållet från denna fil i textfältet under knapparna (se punkt 4 på bilden). Detta sköts av en JavaScript-funktion som läser innehållet i filen med samma namn som texten på knappen man tryckt (se funktionen *readFile(filnamn)*). För att editorn ska kunna hitta filen i den komplexa mappstrukturen som ett GUI kan bestå utav så krävs en sökfunktion (se funktionen *findFile(PathArray, itr, nextFree)*). Denna är också programmerad med JavaScript och är en rekursiv funktion, mer om det i *Kontinuerliga tester och problem*. För att användaren inte ska förlora de ändringar hon gjort i en fil då hon växlar mellan filer så sparas innehållet i textfältet innan innehållet byts ut (se funktionen *writeFile()*).

3- Instruktionsfil:

Instruktioner till varje övning ska gå att läsa när man valt en övning, dessa instruktioner ska man kunna återkomma till utan att förlora något av det man skrivit (se funktionen *showInsructions()*.
Då ett val i dropdown-menyn gjorts så laddas innehållet i filen Instructions.txt in i textfältet. Denna textfil ligger i den valda övningens katalog. Om denna fil ej finns så meddelas detta i textfältet istället. Detta sköts av JavaScript-funktionen *showInsctructions()* som liknar funktionen *readFile(filnamn)* med skillnaden att *showInsructions()* nollställer vissa variabler. Om användaren håller på att arbeta i en kodfil och vill se instruktionerna igen så kan hon trycka på ”show instructions”-knappen. Den kod som användaren höll på att arbeta med sparas innan instruktionerna dyker upp i textfältet.

4- Textfältet:

Innehållet i filerna ska gå att ändra, färgkodning ska ske i realtid:

Detta implementerade vi inte själva utan löstes med hjälp av ett specialiserat textfält kallat *codepress*. Denna finns tillgänglig för gratis nerladdning på http://www.codepress.org/ och har GNU licens (den 18 juli 2007). Mer information om detta specialiserade textfält finns under *Kontinuerliga tester och problem* eller på *codepress* hemsida.

5- Simulatorn:

Koden ska gå att köras i TAT's GUI-simulator med ett knapptryck, de ändringar man gjort i koden ska då visas (se funktionen *runProgram(filename)*).
Detta löstes med JavaScript och ett så kallat ActiveXObject som gör det möjligt att exekvera program som ligger lagrade lokalt på användarens dator. Funktionen som gör detta anropar först funktionen som sparar innehållet i textfältet till filen man arbetar i (se funktionen *writeFile()*). Sedan så exekveras TAT's simulator, den ändrade settings-filen som ändrades av funktionen *listFiles()* tidigare hittar nu rätt filer så att rätt övning laddas in i simulatorn och användaren kan se ett resultat på vad hon kodat.

6- Återställa kod i en fil:

Koden i en övning ska gå att återställa till ursprungligt läge som när man började övningen (se funktionen *resetFile()*).
JavaScript-funktion *resetFile()* återställer den aktuella filen man arbetar med, innehållet i textfältet återställs också. De ändringar man gjort tidigare i filen går då förlorade.

7- Återställa koden i övningens alla filer:

Funktionen *resetAllFiles()* återställer alla filer i hela övningen, editorn hoppar tillbaks till instruktionerna efter den gjort detta.

8- Visa lösning:

Om man vill veta lösningen på en övning så ska det gå att få fram det med hjälp av en knapp som ändrar innehållet i alla kod-filer till den rätta lösningen (se funktionen *showSolution()*).
JavaScript-funktionen *showSolution()* tar reda på vilken fil användaren arbetar i för att sedan ladda in innehållet i motsvarande lösnings-fil till textfältet.

9- Hjälpfil:

TAT-logotypen fungerar som en länk till hjälpfilen till editorn (se funktionen *runProgram(filename)*).
Om en användare är osäker på hur man använder editorn och ej är inloggad på hemsidan så är denna genväg till hjälpfilen en lösning. I hjälpfilen står det kortfattat hur man använder sig utav de olika knapparna.

Hjälpfilen går också att komma åt genom att gå in i editorns huvudkatalog och dubbelklicka med musen på textfilen *Readme.txt*.

10- Uppdatera editorn:

Med ett tryck på denna knapp så kan användaren kolla om hon har den senaste versionen av editorn. Vid knapptryckning så anropar editorn en PHP-fil som kollar mot en databas om editorns versionsnummer är det senaste. När en senare version av editorn lagts upp på servern så ska den ansvarige uppdatera versionsnumret i databasen. Även länken till PHP filen måste uppdateras eftersom det i URL:en skickas med variabler som har information om vilken version editorn har. På sådant sätt kommer en äldre version alltid veta om det finns en nyare version tillgänglig.

3.5.7 Lägga till övning till editorn

TATs simulator måste ha alla filer till ett GUI för att kunna köras. Ett GUI består utav en massa filer och det är endast några få som är intressanta för användaren i en övning. För att lösa detta så måste först alla filer till GUI:t finnas tillgängliga för TATs simulator. Om man sedan har en separat mapp där kopior av de intressanta filerna från GUI:t finns så kan dessa sedan lyftas fram av editorn.
Detta kan lösas om man har en övnings-katalog i editorns rotkatalog. Övnings-katalogen kan heta ”Lessons”. Det är i denna som alla de olika övningarna kommer att ligga och det är där anställda på TAT kan lägga till fler övningar. Varje övning har en egen mapp med sina filer. Alla övningar som befinner sig i Lessons-mappen ska kunna läsas in automatiskt när editorn startas och alla namn på övningarnas mappar kan listas i en dropdown-meny i editor-fönstret (enligt TATs krav 4.3). Detta för att övningarna ska bli lättillgängliga för användaren. För att lägga till en lektion i editorn så ska följande göras:

- Skapa en undermapp i ”Lessons-mappen” med ett passande namn för övningen.
- Kopiera sedan ”resources”-mappen från det GUI man vill använda till den nya övningens mapp man precis skapat. Denna ”resources”-mapp innehåller alla filer som behövs för att TATs simulator ska kunna visa GUI:t.
- Skapa en undermapp i övningens rotkatalog med namnet ”Show”. Det är o denna mapp som alla intressanta filer för övningen ska finnas.
- Kopiera från resources-mappen de filer man vill göra tillgängliga i övningen till ”Show”-mappen.
- Vill man lägga till instruktioner till användaren så skapar man ett textdokument med namnet ”Instructions.txt” i övningens rotkatalog. Detta dokument ska laddas in automatiskt av editorn då användaren väljer övningen.

- Vill man att användaren ska kunna se lösningen på övningen så kan man skapa en ”Solution”-mapp i övningens rotkatalog där man kopierar in lösningsfilerna.

Dessa steg ska följas för att lägga till en övning till editorn. Gör man fel på något av dessa fel så ska editorn helst inte sluta fungera utan meddela användaren om vad problemet är. Två krav har vi lagt till som gör det möjligt för editorn att varna om övningsfiler saknas eller instruktionsfil.

3.5.8 Problem under utveckling av editorn och lösning.

Under utvecklingens gång så har kontinuerliga tester pågått. Så fort en av editorns funktioner implementerats så har denna testats i editorprototypen. Som med all utveckling av datorprogram så har mindre problem uppstått som man sedan rättat till. I detta avsnitt så ska inte dessa småproblem tas upp, utan de lite större och mer intressanta problemen som dykt upp. Dessa problem har fått oss att inse att programmet vi utvecklat visat sig vara mer komplext än vi från början tänkt oss.

Sökfunktion:

Ett av dessa större problem har varit att editorn måste söka igenom övningarnas komplexa mappstrukturer för att finna de filer som en övning bearbetar. Ett av TATs GUI:n kan bestå utav upp till hundra filer i ett tjugotal olika kataloger och underkataloger.

JavaScript har en inbyggd funktion som kan söka efter en fil i en katalog men ej dess underkataloger. Här är ett exempel på en mappstruktur för att illustrera JavaScripts sökfunktions kapacitet:

C:\huvudmapp\
C:\huvudmapp\undermapp1\
C:\huvudmapp\undermapp2\
C:\huvudmapp\fil1.txt
C:\huvudmapp\undermapp1\fil2.txt

JavaScripts sökfunktion kan hitta fil1.txt om man anropar funktionen med huvudmappen som parameter. Men den kan inte hitta fil2.txt, utan man måste ange den direkta sökvägen till undermapp1.
Detta är problematiskt eftersom många av de önskvärda filerna befinner sig oftast djupt nere i katalogstrukturen. För att lösa detta så har vi implementerat en egen sökfunktion utifrån den sökfunktion JavaScript erbjuder.
Vår nya funktion söker igenom alla filer i en katalog precis som den vanliga JavaScript sökfunktionen. Men därefter anropar den sig själv för varje underkatalog i den första katalogen och sedan för varje underkatalog i dessa underkataloger osv. En funktion som anropar sig själv kallas för en rekursiv

funktion och kräver en del förarbete med papper och penna innan den ska implementeras för att bli rätt. Detta på grund av det är väldigt enkelt för en rekursiv funktion att hamna i en oändlighets-loop, vilket innebär att den aldrig blir klar med sin uppgift.

För att sökfunktionen ska hålla reda på hur många undermappar den ska söka igenom måste varje undermapps sökväg lagras i en vektor. När sökfunktionen för första gången går igenom huvudkatalogen så söker den både igenom filerna och sparar alla sökvägar till dess undermappar i vektorn. När den sedan sökt igenom huvudkatalogen så anropar den sökfunktionen igen fast nu med den första lagrade sökvägen i vektorn som parameter. Här är ett exempel på mappstruktur för att illustrera hur det går till:

C:\mappA\
C:\mappA\mappA1\
C:\mappA\mappA2\
C:\mappA\mappA3\
C:\mappA\mappA3\mappA3a
C:\mappA\mappA3\mappA3b
C:\mappB\
C:\mappB\mappB1\
C:\mappB\mappB2\
C:\mappB\mappB2\mappB2a

Om vår funktion anropas med C:\ som sökväg så kommer den att lagra alla mappar på C:\ i vektorn. Dessa är mappA och mappB. När den sedan är klar så anropar den sig själv med den första lagrade sökvägen i vektorn, dvs mappA. Nu kommer den att söka igenom mappA och lagra alla undermappar i den, dvs mappA1, mappA2 och mappA3. När den är klar med detta så anropar den sig själv med nästa sökväg i vektorn som nu är mappB, här finns mapparna mappB1 och mappB2. Detta fortsätter ända tills den kommit till mappA3b som blir den sista mapp som söks igenom. Eftersom det inte finns några fler mappar i den så avslutas funktionen.

Färgkodningen:
Ett av de viktigaste kraven för editorn ställda av TAT var att editorns textfält skulle ha färgkodning. Detta innebär att de olika elementen i koden skulle färgas olika beroende på funktion. Här nedan är ett exempel på färgkodad kod:

```
<container id=”window” width=”100” height=”300”>
```

Att ha färgkodad kod gör det mycket enklare för kodaren att uppmärksamma olika delar av koden eller t.ex. värden som den rosa texten.

Det var färgkodningen tillsammans med säkerhetsrisker med att köra program som avgjorde vilket programmeringsspråk vi valde till editorn.
Färgkodning kräver en ganska avancerad lösning och denna lösning fanns redan på Internet i form av ett skript kodat i JavaScript. Detta skript heter codepress och finns tillgängligt på www.codepress.org. Skriptet är gratis för användning och en god dokumetation om hur det fungerar fanns tillgänglig. Codepress-skriptet består av ett flertal filer och stöder ett stort antal olika programmerings-språk. Detta är bra eftersom olika programmeringsspråk kräver olika sätt att färgas på. TATs egna programmeringsspråk som inte är så välkänt fanns såklart inte men eftersom deras språk är baserad på XML-språket som är betydligt mer spritt så använde vi oss utav XML-inställningarna. Codepress inställningarna för XML kodningens färger tyckte vi inte var så bra, så dessa ändrades till andra färger. Dessa färger ändrades genom att ändra i den CSS-fil som XML färgkodningen använder.

4 Resultat och Analys

4.1 Resultat

Applikationen med handledningstillfällen som skapades till TAT och deras programmeringsspråk Cascades finns att hitta på:
http://www.jonathan-stahl.se/1EXJOBB/login.php
På denna sida finns även en länk till editorn där denna kan laddas hem. En förutsättning för att ladda hem editorn är att man först loggat in. Login och lösenord kan fås genom att maila: ihm04pno@student.lu.se eller jojostahl@gmail.com .

För att återknyta till syftet med projektet så har en applikation, i form av en hemsida där det finns möjlighet att lagra handledningstillfällen och tillhörande editor för nyanställda och kunder till TAT skapats. Här ska TATs XML-baserade programmeringsspråk TAT Cascades gå att lära sig. Hur väl applikationen som skapats överensstämmer med det förväntade resultatet belyses i nästa kapitel, 4.2: Granskning av kravspecifikation.

4.2 Granskning av kravspecifikation

När vi har vårt färdiga resultat, så behöver detta jämföras med vad som krävdes av applikationen. Vid en genomgång av kravspecifikationen så stämmer en punkt inte överens med den klara applikationen. Det är ett grönmarkerat krav vilket innebär arr det är ett krav från TAT.

> Krav 3.3- Editorn ska innehålla övningar som motsvarar alla kapitel som finns på hemsidan.

Detta krav har inte uppfyllts eftersom det bestämdes att innehåll ej skulle skapas till editor och hemsida. Dock är både hemsida och editor skapad så att det finns möjlighet att lägga in övningar och innehåll.

Det är endast på denna punkt applikationen avviker från kravspecifikationen, i övrigt stämmer allt.

4.3 Granskning av arbetsmetodiken

De första två månaderna av arbetet utfördes på TATs kontor i Malmö. Här bestämdes vad som skulle göras, hur applikationen skulle se ut och huvudfunktionerna av applikation och editor implementerades. TAT expanderas väldigt mycket för tillfället och efter dessa två månader så hade dem inte plats för examensarbetare i sina lokaler utan andra projektgrupper inom TAT behövde dessa platser då personal från Korea och USA anlände.

Under projektets gång har två tidsplaneringar följts. Eftersom arbetet kom igång ganska sent (slutet på mars) gjorde vi en medvetet ganska tidsoptimistisk tidsplanering i vilken vi redovisade vårt examensarbete i början på juni. I denna planering var endast en vecka planerad till rapportskrivning. Denna tidsplanering klarade vi inte av att hålla utan en ny gjordes där rapport skrevs under sommaren. Metodiken för utveckling av produkten och samarbetet med TAT under perioden vi satt på deras kontor, fungerade mycket bra. Samtliga deadlines rörande utveckling av applikation som metodiken beskrivit hölls, från granskning av kravspecifikation till implementering. Designen av hemsida och editor gick fortare än beräknat eftersom inga krav lades på applikationens utseende och en simpel, men funktionell design skapades.
Trots detta sprängdes även den andra tidsplaneringen. Detta förklaras med att nästan 3 veckor lades på att skriva innehåll till handledningsövningarna. Efter att ha arbetat med projektet i över en månad bestämdes det att innehållet till handledningstillfällena inte skulle vara en del av applikationen.
Vidare ville TAT att applikationen skulle bli klar innan semestern började vilket var 1månad innan vi planerat att ha den klar. Detta resulterade i att all rapportskrivning lades åt sidan och fokus lades på att få applikationen klar. Den 13:e juni presenterades applikationen på TAT. Veckan efter detta lades på nödvändiga kompletteringar och vecka 25 var allt tekniskt arbete klart (3veckor tidigare än planerat). Mer om presentationen och vad TAT tyckte om resultatet finns att läsa om i slutsatsen.

Efter att applikationen blivit helt klar fanns fyra veckor kvar till arbetet enligt planeringen skulle vara helt klar. Efter att dessa fyra veckor gått var vi inte klara. Detta förklaras med att rapporten är ganska tung och tagit mycket längre tid än beräknat.

4.4 Granskning av implementering och alternativa lösningar

I detta avsnitt beskrivs alternativa lösningar på problem som uppfyller kraven från kravspecifikationen, men som vi tycker kunde ha fungerat bättre om man modifierat dem en del.

5.4.1 Blockhantering:
Addering och editering av olika block fungerar, men man behöver ha en viss kunskap i hur alltsammans hänger i hop. När t.ex. en bild i nuläget läggs till i ett bildblock så skrivs endast namnet på filen in, sen visas bilden. Detta kräver att bilden redan ligger på servern. Funktionaliteten av detta kan diskuteras. Det hade fungerat bättre med även en ”upload” knapp där man kunnat ladda upp bilderna, så man slipper ladda upp bilden manuellt via ett ftp program eller liknande.

Det hade även varit bra om där under textarean som ska innehålla bilden ligger en textarea till, där text kan skrivas. Dessa två textareor skulle sedan läggas ihop

och representeras som ett block. Det är sällan man endast vill lägga in endast en bild, eller kodblock. I dem flesta fall vill man även skriva in ett par rader text. För tillfället så lägger man in en bild, trycker ok, trycker edit på samma block och lägger sedan in texten.

Utöver detta borde de olika blockspecifika taggarna (<img><textarea>) adderas under tiden som sidan ritas upp, inte ligga i databasen som dem gör nu. Om dem ligger i databasen som nu så kan man trycka på edit och sedan ta bort html taggarna vilket medför att blocket inte visas på det sätt det ska. Detta medför även att vid addering av kodblock behövs en dropdownlista där man kan skriva in hur många rader man vill att kodfönstret ska innehålla. Nuvarande lösning fungerar, men en felsäker och lite smidigare lösning hade varit att föredra.

4.5 Erfarenheter från examensarbetet

Under projektets gång har en del olika erfarenheter och kunskaper skapats. Vad gäller den tekniska biten så har vi lärt oss PHP, SQL och JavaScript betydligt bättre än de kunskaper vi hade innan där vi endast skrapat lite på ytan inom dessa områden. Rapportskrivandet har gett stor insikt i hur rapportskrivning på denna nivå går till och hade arbetet gjorts om en gång till från början, så hade mycket mer tid lagts på att skriva rapport parallellt med övrigt arbete. Under två veckor av projektets gång skrevs rapport samtidigt som utveckling av applikationen pågick, rapportskrivandet borde ha påbörjats långt tidigare.

Hade vi fått en ny chans att göra en applikation för utlärning av ett programmeringsspråk så hade vi nog kollat mer på programmeringsspråket asp.net. Vi skulle se om det fanns bättre möjligheter att integrera en editor på hemsidan. Fördelarna med en inbyggd editor hade varit att övningarna skulle vara mer tillgängliga för både användarna och administratörerna. Användarna hade kunnat gå igenom lektioner med små övningar där kod hade kunnat testas direkt utan att tappa tråden i avsnittet man läste om. Problemet med detta i vårt fall var att TATs simulator inte gick att köras via hemsidan. Men detta kanske inte är fallet för ett annat programmeringsspråk med en annan simulator.

4.6 Kommentarer på applikationen från nyanställd på TAT

En befogad fråga när hemsidan som ska innehålla handledningstillfällena och editorn är klara blir givetvis om vad som producerats är bra? Minskas inlärningstiden för Cascades när en nyanställd eller kund ska lära sig detta programmeringsspråk? Detta är svårt att veta eftersom där inte finns någon anställd på TAT som skapar och lägger handledningstillfällen och övningar till dessa.

Under tiden som arbetet gjorts så har en ny Cascades kodare anställts och vi förklarade för honom hur det var tänkt att ramverket skulle fungera. Sedan bad vi honom kommentera hur han såg på vad som skapats.
Han tyckte att Motion Lab som är en editor till Cascades utvecklad av TAT själva, var en betydligt bättre editor att arbeta i eftersom den är skräddarsydd till Cascades och där finns många bra funktioner. I denna editor finns färdiga komponenter som man kan använda i sitt gui plus att där är en exempel suite med en mängd exempel på olika GUIs. Dessa gränssnitt är klara GUIn som man kan titta i och där finns i Motion Lab likt i våran editor en knapp för att starta en simulator som visualiserar vad koden gör. Detta tyckte han medförde att man mest kopierade och klistrade in befintlig kod och sällan eller aldrig skapade något ny egen kod. Detta gjorde att man lätt missade förståelsen och strukturen för hur ett GUI i Cascades borde byggas upp. Tanken med uppgifter som ska lösas var bra eftersom man får försöka lösa dessa på egen hand och inte ta klara lösningar från existerande GUIn.

Vidare skulle han gärna sett tutorials som går igenom enklare GUIs och förklarar allt eftersom det i nuläget inte finns alls. Han tror även att detta skulle kunna användas vid små work shops för kunder som ska introduceras till Cascades innan de börjar använda Motion Lab.

5 Slutsats

Vid mötet den 13:e juni där applikationen presenterades för TAT diskuterades vad som skulle hända med produkten framöver. Alla var överens om att applikationen var bra, speciellt den skyddade miljö-aspekten . Att laborera i en skyddad miljö som man gör i editorn, där man lätt kan återställa kodfilerna till sitt ursprungstillstånd tyckte Cascadekodarna var väldigt bra för en ovan användare.

Vidare diskuterades hur applikationen som helhet ska komma att användas. Att ha den i Motion Lab var uteslutet eftersom programmerarna där redan sitter på väldigt tätt åtsittande deadlines. Däremot kommer förmodligen hemsidan att byggas in på TATs intranät. Editorn kommer att föreslås till nyanställda som ett kompletterande läromedel till dokumentationen om TAT får tid att skapa kompletta övningar till den. Med tanke på TATs expansiva stadium så kan vi tänka oss att ett bättre inlärningsmedel kommer att utvecklas och då hoppas vi att våran applikation kommer att tas i åtanke och inspirera dem till ett bra sådant.

Appendix A – TAT Tutorials kravspecifikation

Här presenteras kravspecifikationen i dess helhet. Denna innehåller både TAT's krav (grönmarkerade) och de krav som vi kommit fram till.

1-Introduktion

1.1-Syfte

Syftet är att skapa ett läromedel för nyanställda och kunder där The Astonishing Tribes XML-baserade programmeringsspråk Cascades och Kastor går att lära sig.

1.2-Mål

Läromedlet som skapas skall bli ett interaktivt läromedel i form av en hemsida där handledningstillfällen kommer att ges tillsammans med uppgifter som kan lösas i medföljande editor.

2-Krav på hemsidan

2.1- För att editorn skall gå att använda så behövs ett kompletterande ramverk, detta ramverk görs i form av en hemsida.

2.2- På hemsidan finns handledningstillfällen.

2.2.1- Varje handledningstillfälle är uppdelat i sektioner som kallas kapitel.

2.3- Hemsidan skall använda sig av ett inloggningssystem så att obehöriga ej kan ta del av utbildningsmaterialet.

2.3.1- Inloggningssystemet skall ha användarnamn och lösenord.

2.3.2- Det skall finnas administratörskonton och användarkonton.

2.3.3- Endast administratörskonton kan förändra block.

2.4- Språket på hemsidan skall vara engelska.

2.5- Hemsidan skall vara anpassad för bredbandanvändare med en kapacitet av minst 512kBit/sekund.

2.6- Hemsidan kommer vara optimerad för Microsofts Internet Explorer.

2.7- Sidan skall göras för upplösningar i 1024*768 och större.

3- Krav på editorn

3.1- Editorn ska vara en fristående applikation som öppnas i ett nytt fönster.

3.1.1- Editorn ska gå att ladda ner från hemsidan.

3.2- Editorn ska färgkoda koden i filerna enligt XML standard.

3.2.1- Färgkodningen ska ske i realtid.

3.3- Editorn ska innehålla övningar som motsvarar alla kapitel som finns på hemsidan.

3.3.1- Övningar i editorn ska ha samma namn som övningstillfällena på hemsidan, inklusive nummer.

3.4- I editorn skall det finnas en uppdatera knapp, på vilken man kan kontrollera mot en server om det finns en nyare version av editorn.

3.5- Editorn ska ha minst en kodfil att visa upp för varje Övning.

3.5.1- Alla kodfiler i en övning ska gå att återställa till ursprungsläget genom att trycka på en reset knapp.

3.6- Koden ska gå att testköra när som helst med hjälp av ett tryck på Run Simulator-knappen.

3.7 Det ska gå att lägga till övningar till editorn.

3.8- Språket i editorn ska vara engelska.

3.9- Det ska finnas en hjälpfil till editorn som förklarar grunderna i användandet av editorn.

3.9.1- Hjälpfilen till TAT-anställda ska förklara hur man lägger till en övning till editorn, instruktioner samt filer.

3.10- Editorn ska inte överstiga en filstorlek på cirka 15mb.

Appendix B - Ordlista

Här listas alla ord, förkortningar och begrepp som kan vara nya för läsaren.

CSS – Cascading Style Sheets:
Programmeringsspråk som används för att beskriva hur information på en webbsida ska presenteras.

DBMS – Databashanterare:
En databashanterare är en mjukvara för att hantera data i en databas. T.ex. MySQL eller Oracle.

GUI – Graphical User Interface:
Användargränssnitt, i detta fall vad man ser och interagerar med på en mobiltelefons skärm.

HTA – HTML Application:
Är en Microsoft Windows applikation skriven i HTML utan de restriktioner HTML har i en Internetläsare. Applikationen körs lokalt.

HTML – Hyper Text Markup Language:
Programmeringsspråk som används för att skapa hemsidor.

PHP – PHP Hypertext Preprocessor:
Kraftfullt programmeringsspråk som ligger på servern. Används för att skapa dynamiska och interaktiva webbsidor.

URL – Uniform Rescource Locator:
URL är den korrekta benämningen för en webbadress. Den talar om vilket protokoll som ska användas, vilken domän man ska leta sig fram till, vilken plats och fil samt vilken port som ska användas.

SQL - Structured Query Language:
SQL är ett programmeringsspråk som är designat för att skapa tabeller(relationer) samt att hämta och sköta data från en databashanterare (DBMS) på olika sätt.

XML – eXtensible Markup Language: är ett universellt och utbyggbart märkspråk.

Appendix C – Kodfil till editorn

Om ni vill ha den så fråga oss.
Mail: jojostahl@gmail.com