Minneshantering för iPhone

Begränsning eller möjlighet?

BASHAR SHAYA
och GUSTAV JOHANSSON

KTH Datavetenskap
och kommunikation

Examensarbete
Stockholm, Sverige 2010

Minneshantering för iPhone

Begränsning eller möjlighet?

B A S H A R S H A Y A
o c h G U S T A V J O H A N S S O N

Examensarbete i datalogi om 15 högskolepoäng
vid Programmet för datateknik
Kungliga Tekniska Högskolan år 2010
Handledare på CSC var Cristian Bogdan
Examinator var Mads Dam

URL: www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2010/shaya_bashar_OCH_johansson_gustav_K10040.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Sammanfattning

När iPhone släpptes för försäljning den 9e januari 2007 började en världsledande epok inom mobil- och smartphonebranschen. Apple lanserade senare samma år API för utveckling av programvara till iPhone. En helt ny värld blev tillgänglig för programmerare och därmed söktes kunskap och erfarenhet, vilket gav upphov till att iPhone blev den telefon man skulle programmera och utveckla för. För att bli en bra programmerare till iPhone måste man kunna hantera ett flertal viktiga aspekter. Det finns en begränsning på CPU, batteri men framför allt minne. Minnesbegränsningar är en stor och viktig del i iPhone-programmering. Man har ingen skräpsamlare i Objective-C, vilket är språket som används för utveckling av programvara för iPhone. Detta gör att man som programmerare måste hantera all minnesallokering på egen hand. För att skapa en variabel måste man allokera plats åt den för att sedan initiera den. Eftersom det är mycket lätt att program kraschar pga. felhantering av minnen och allokering, är det viktigt att man som programmerare är mycket uppmärksam på just allokeringen. Vi hoppas att man som programmerare blir nyfiken och intresserad av att börja programmera för iPhone, och samtidigt får nytta av den samling av information som vi har gjort på viktiga områden med fokus på minneshantering.

Abstract

Memory Management for iPhone: Limitation or opportunity?

When the iPhone was released for the market on January the 9th, 2007, a new era began in the ongoing and developing world of mobile- and smartphones. Later that year Apple introduced the API for the iPhone, which opened up doors for a whole new world of programmers and suddenly the iPhone was the phone to develop for. To become a good programmer for the iPhone, you have to deal with and have in mind several important aspects. The telephone has a limitation on the CPU, battery and especially the memory. The memory limitations are a large and important part of the iPhone programming. There is no garbage collector in Objective-C, which is the language used for developing software for the iPhone. This means that you have to deal with all kinds of memory allocation of one's own accord when programming. To create a variable you need to allocate space for it, and then initialize it. Since program and system crashes often occurs, the mishandling of memory allocation makes that programmers must be very aware of the allocation. We hope that programmers will be interested and anxious to begin programming for the iPhone and that they benefit from this report, which has a wide collection of data that we have made for some key areas and especially with focus on memory management.

Förord

Denna rapport har gjorts i syfte att studera hur samt om minneshantering vid programmering av applikationer för en iPhone begränsar eller skapar möjligheter att utvecklas. Då en del av termerna ej har eller är svåra att översätta från engelska till svenska, har vi valt att behålla ursprungsnamnet. Arbetet, Minneshantering för iPhone, har gjorts av Bashar Shaya och Gustav Johansson, båda studenter och läser sitt tredje år vid skolan CSC vid Kungliga Tekniska högskolan i Stockholm, inom datalogi med Mats Dam som examinator. Arbetet har fördelats på en jämn nivå så att båda har fått lika mycket att arbeta med.

Bashar Shaya har skrivit: Inledning, Bakgrund, State of Art, Historia om Mac OS X SDK, utvecklingsverktyg samt Hur man utvecklar i Xcode.

Gustav Johansson har skrivit: iPhone; Prestanda och grafik, Begränsningar samt utvecklingsverktyg och utvecklingsmöjligheter.

Tillsammans har vi jobbat med fördjupningen inom minneshantering, där Gustav har koncentrerat sig på felhanteringen och varför den uppstår och där Bashar har haft fokus på allokering och hur det fungerar. Vi har även skrivit diskussionen och slutsatsen tillsammans för att få en så bra avrundning som möjligt av hela arbetet.

Innehållsförteckning

Innehåll

1 Inledning

Den 9e januari 2007, under Apples årliga konferens, meddelade man att mobiltelefonin skulle tas till nya höjder. Med stort intresse bevakades detta från världens alla hörn.
En ny mobiltelefon, senare kallad iPhone, såg dagens ljus[1]. Denna iPhone kom att revolutionera och inspirera mobiltelefonin och hela den tekniska världen. Lanseringen av denna telefon kom från att man ville utveckla Personal Digital Assistant(PDA) användandet. Vad vi är intresserade av är hur denna telefon kom att bli just detta avstamp i dagens samhälle!

För att kunna beskriva framgången av iPhone måste vi gå tillbaka till hur mobiltelefonin såg ut innan 2007. Vilken standard var det som användes då? Hur såg försäljningen ut och vad var populärast bland konsumenterna? Därför har även bakgrunden till iPhone tagits med i rapporten.

En av de viktigaste aspekterna och där rapportens fokus ligger på är just; *Hur påverkas minnet och minneshanteringen och hur begränsas en applikationen av detta?*
Detta är bland annat en av frågorna vi har ställt oss i denna rapport, utöver en rad andra frågor som inte bara berör programmeringen utan även utvecklingen och påverkan av denna telefon.

Varför lyckades inte Windows mobile på samma sätt som iPhone har gjort och hur skiljer sig dessa operativsystem åt? Dagens utveckling och distribution av applikationer till iPhone är stor, närmare 140 000 stycken applikationer[2], men hur hamnade vi här och vart är vi på väg? Dessa var några av anledningarna som ledde in oss på ämnet och det var det som drev oss till att skriva denna rapport.

Som det ser ut i dagens samhälle och främst i Sverige har iPhone verkligen slagit igenom med en väldig kraft och detta har även påverkat de andra aktörerna inom mobilbranschen. Påverkan har skett genom att man har satt upp höga mål och förväntningar för mobiltillverkning, och då syftar vi inte bara om design utan även om kvalitet och känsla. Enligt färska rapporter har iPhone blivit en genuin och alldaglig ”Svensson-telefon som verkligen har slagit igenom och gått hem bland svenska folket”.[3] Var man än tittar, på tunnelbanan, på bussen eller ute på våra gator, har iPhone blivit en trendsättare och ikon, enligt media. Detta för tankarna tillbaka till en annan känd trend från företaget Apple. Känner ni inte igen detta mönster och beteende som uppstod i början på 2000-talet? Jo, detta kan kännas igen i att det var iPod som lyftes fram och kallade på samma uppmärksamhet som dagens iPhone har gjort. Den blev en verklig ikon och de vita hörlurarna syntes i princip överallt!

Hur har denna typ av effekt, det stora genomslaget av en ny telefon, påverkat oss som människor, men inte allra minst de programmerare som jobbar med att utveckla mjukvara? *Hur påverkas minnet och minneshanteringen och hur begränsas en applikationen av detta?*

[1] http://en.wikipedia.org/w/index.php?title=History_of_the_iPhone&oldid=350627255
[2] http://www.apple.com/ipad/app-store/
[3] http://www.e24.se/business/it-och-telekom/iphone-svenssons-nya-mobiltelefon_1865013.e24

Det är nämligen så att intresset för att skapa program, eller applikationer, för mobiltelefoner och smartphones har ökat de senaste åren.[4] Varför och hur har det gjort det är en del av den problemformulering som vi tänker besvara i denna rapport.

Vi kommer även att försöka besvara följande frågeställningar:

Hur använder programmeraren utvecklingsverktygen när han/hon skapar applikationer till telefonen?

vilken kan ses som en förgrening till huvudfrågan vilken är;

- *Hur påverkas minnet och minneshanteringen och hur begränsas en applikationen av detta?*

Disposition

Här nedan beskrivs de olika delarna av rapporten och hur den är uppbyggd.

Del 1
Tar upp bakgrunden till iPhone OS för att ge läsaren en bra inledning och översikt för att kunna skapa en bild av det som komma skall.

Del 2
Kommer att beskriva de olika delarna som berör iPhone. Här behandlas de olika utvecklingsverktygen, den miljö som man arbetar, gällande både iPhone och OS X och sedan även lite historia om OS X, iPhone och Software Development Kit (SDK). Vi kommer att börja med att ge en allmän inledning till de olika utvecklingsverktygen och miljöerna, för att senare gå över till mer specifik programmering för just iPhone.

Del 3
Här kommer diskussion och slutsats avsluta rapporten och knyta samman det som har skrivits. Här besvaras även rapportens inledande frågeställningar och problemformuleringen.

[4] http://www.treetop.se/sv/Nyheter/Treetop-CV

Del 1

2 Bakgrund

Redan från första början, då den första iPhone telefonen lanserades, den 9 januari 2007, har Apple varit världsledande, inte bara inom mobiltelefonutveckling, utan inom flera områden såsom datorer och musiktillbehör. Idag, tre år senare, har det sålts miljontals telefoner och försäljningen antas öka under de kommande åren.[5]

Vad är då hemligheten bakom denna telefon? Vad har gjort den till en ledande ikon i dagens tekniksamhälle och hur kan den komma att utvecklas?

De flesta ledande mobiltelefontillverkarna försöker ta efter Apples geniala gränssnitt och stilrena design. Men vad är det som har givit iPhone dess rykte som sedan har fört den till en enorm framgång bland världens alla användare?

Själva bakgrunden till hur och varför man kom på idén att lansera en ny telefon från Apple började redan i mitten på 1990-talet. Det hela började när man lanserade en PDA från Apple, även kallad ”The Newton”.[6] Denna lansering genomfördes efter att man hade märkt att det fanns en efterfrågan på dessa mobila apparater, som man inte bara kunde ringa med utan som man även kunde utföra andra viktiga sysslor med, exempelvis organisera kalendern och telefonboken.

Redan 2003 valde man att inte fortsätta tillverkningen av PDA och man avslutade även forskningen och utvecklingen av dessa. Man insåg att mobiltelefonindustrin var framtiden. Man behövde en mobiltelefon som hade allt och som kunde synkronisera med resten av de nödvändiga applikationerna, såsom kalender, kunna köpa och lagra media direkt från iTunes och samtidigt kunna använda den som en vanlig men ändå en bra telefon.

Apple inledde ett samarbete med Motorola och lanserade den första telefonen, ROKR E1.[7] Det som utmärkte denna telefon var att man som användare kunde köpa musik direkt från iTunes och lagra detta direkt i telefonens minne eller i ett Secure Digital-kort. Samarbetet med Motorola var något som Apple inte var helt nöjda med, då Motorola inte direkt gick i samma tankebanor som Apple. Motorola använde sig av annorlunda teknik och design. I september 2006 avslutade man tillverkningen av denna produkt och släppte en ny version av iTunes där man läckte information om att en ny telefon som skulle kunna hantera bilder och video. Några månader senare, 9e januari 2007, släppte man iPhone, som beskrevs som den nya generationens telefon.[8] Sedan dess har framgångarna bara fortsatt, med efterföljarna iPhone 3G och 3GS.

[5] http://www.e24.se/lifestyle/prylar/iphone-ett-lyft-for-apple-igen_1820489.e24
[6] http://en.wikipedia.org/wiki/History_of_the_iPhone
[7] http://www.motorola.com/motoinfo/product/details.jsp?globalObjectId=117
[8] http://www.apple.com/pr/library/2007/01/09iphone.html

Del 2

3 State of Art

3.1 Windows Mobile

Apple är den största ”nya generationens mobiltillverkare”. Men många andra företag vill ta efter succén som Apple skapade och tog över efter Windows Mobile.
Idag har vi en stor marknad med Android telefoner, vilket är operativsystemet som skapats och utvecklats av Google.[9] Även på senare år har Windows mobiles utvecklats mycket men gapet mellan Apple och Windows har fortsatt att öka[10]. Windows Mobile med Microsoft i spetsen, var en av de större aktörerna först ut på marknaden, har förlorat en hel del marknadsandelar i jämförelse med iPhone OS och Android.

Vad är Windows Mobile? De flesta människor har suttit framför en PC och har då haft Windows som operativsystem. Nu har Microsoft även gett sig in i mobilbranschen och man har lanserat det som heter Windows Mobile. Som det låter på namnet är detta operativsystem för mobila enheter såsom mobiltelefoner, PDA och smartphones. Själva designen och systemet bygger på Windows operativsystem för datorer, men här har man skalat ner det och gjort det anpassningsbart för mobiler. Det man var snabb med, till skillnad från andra aktörer, var att olika tredjepartstillverkare fick licens att släppa applikationer på Microsofts egen marknad, Windows Marketplace for Mobile. Det hela började med versionen Pocket PC2000 och Microsoft har efter det släppt nya versioner nästan varje år. De flesta versionerna bygger på storebror, Windows OS för PC, vilket har lett till att man har släppt många uppdateringar av systemet.

Fastän Windows Mobile för några år sedan var det främsta av de tre ovannämnda operativsystemen, har Microsoft tappat en stor andel, nästan en femtedel, av marknaden på senare år.[11] Det man har förlorat, jämfört med iPhone OS och Android, är främst applikationsmarknaden. Microsoft har inte alls lanserat så många applikationer som de två andra operativsystemen och även Microsofts marketplace för nerladdning av applikationer har inte heller den slagit igenom som exempelvis Apples Appstore har gjort.

3.1.1 Windows mobile SDK

Det finns olika tillvägagångssätt för programmerare som vill skriva applikationer för Windows Mobile. Man kan använda sig utav de tre alternativ som finns beskrivna nedan:

- Skriva kod i Visual C++
- Managed Code, vilken fungerar med .NET Compact Framework[12]
- Server-Side Code som kan användas med Internet Explorer Mobile eller en mobil klient som används på enheten

[9] http://developer.android.com/guide/basics/what-is-android.html
[10] http://ohsohightech.se/ar-iphone-det-nya-windows/
[11] http://arstechnica.com/microsoft/news/2009/08/windows-mobile-loses-27-of-smartphone-market-in-q2.ars
[12] http://msdn.microsoft.com/en-us/library/2weec7k5.aspx

Microsoft brukar vanligtvis släppa SDK som kan samarbeta med deras Visual Studio utvecklingsmiljö. Dessa SDK innehåller ”*bilder*” eller ”*images*” av emulatorer som underlättar och hjälper utvecklarna av applikationer. Några av sakerna som programmerarna kan få hjälp med är felsökning och körning medan man skriver applikationer.

Något som Microsoft har tänkt på för att locka unga programmerare är att man har släppt Visual Studio 2008/2005 Professional Edition gratis för alla studenter. På detta sätt hoppas man få upp intresset för Windows Mobile och utvecklingen av applikationer för den Microsofts marknad.

3.2 Android – Operating system

Android är världens första mobila operativsystem som är helt öppet, det vill säga med öppen källkod. Nyligen bestämde sig 34 stycken ledande företag inom mobil- och teknikbranschen för att skapa en allians för framtagandet av ett mer öppet och bättre operativsystem för mobiltelefoner. Den femte november 2007 presenterade dessa innovatörer Android[13].

“*Building a better mobile phone would enrich the lives of countless people across the globe*”, skriver Open Handset Alliance (OHA) på sin hemsida.[14] Att bygga en bättre och mer öppen plattform var ett primärt mål för skapandet av alliansen. Men det fanns även ett behov av att skapa en gemensam förening för att kunna konkurrera med de två mer kända och konkurrenskraftiga på marknaden för operativsystem, Microsoft och Apple. OHA består idag av 65 ledande IT- och telefonföretag med Google i spetsen. För att bli medlem i OHA krävs att man på ett eller annat sätt bidrar till utvecklingen av operativsystemet Android.

Android byggdes inledningsvis för att möjliggöra för utvecklare att skapa fängslande och attraktiva mobila applikationer som till fullo utnyttjar allt som kan krävas av en ”handenhet”. Android byggdes i själva verket för att vara öppet och man kan genom anrop komma åt telefonens grundläggande funktioner såsom att ringa, ta bilder med kameran, nå telefonboken med mera. Allt detta för att göra det enkelt för användare att bygga rikare och mer underhållande applikationer.

Android bygger på en öppen Linux-kärna. Dessutom använder man en egen virtuell maskin, Dalvik, som är utformat för att optimera minne och hårdvara i den mobila miljön.
Den öppna källkoden skapar möjligheter och öppnar dörrar för framtida teknik och utformning. Plattformen kommer att, tillsammans med vanliga användare, utvecklas för mer innovativa lösningar.

Operativsystemet Android har funnits som öppen källkod sedan 21a oktober 2008.
SDK är skrivet i java och finns till Windows, Mac OS X och Linux under en så kallad Apache Licens. Denna licens kräver bevarande av upphovsrätten. Man kan använda källkoden för utveckling av proprietär programvara samt fri och öppen programvara.[15] Det har släppts tre större uppdateringar och den senaste versionen, 2.1, släpptes den 12 januari 2010.

[13] http://www.android.com/about/timeline.html
[14] http://www.openhandsetalliance.com/oha_overview.html
[15] http://www.apache.org/licenses/

3.2.1 Android SDK

Med denna SDK kan utvecklare använda exempelvis programmet Eclipse för att skriva applikationer. I denna utvecklingsmiljö utförs arbetet även här som i Windows Mobile, med managed code i språket Java. Det man egentligen gör är att man skriver managed code, och denna skrivs i Java med bibliotek som är utvecklat av Google.

Själva SDK innehåller en omfattande del av utvecklingsverktygen såsom; Felsöknings bibliotek, emulator, dokumentation, exempelkod och guider. I dagsläget har man stöd för de flesta systemen, däribland Windows OS, Mac OS X och Linux.

Den 12e november 2007 släppte man en förhandstitt av Googles SDK, men det dröjde ända till den 23e september 2008 innan man fick ut version 1 av detta SDK. I denna version hade man åtgärdat några buggar och lagt till några små tillägg. I dagsläget är man på version 2.1 och version 2.5 väntas komma till sommaren 2010.

3.3 Blackberry OS (RIM)

Blackberry OS, RIM, operativsystem är det absolut största i USA och hamnar på andra plats internationellt sett efter Nokias operativsystem Symbian. [16] Blackberry OS har utvecklats utav det kanadensiska företaget Research In Motion (RIM). Precis som med en smartphone ingår de vanligaste applikationerna, såsom telefonbok, kalender och anteckningar, men utöver dessa finns den mest kända; funktionen som möjliggör att man kan ta emot och skicka email. Detta kan göras så fort man har kontakt med en operatör eller ett trådlöst nätverk.

I mitten på 1990-talet förändrades hela affärsvärlden och det söktes efter nya sätt att kommunicera på. Användarna behövde snabbt ta beslut och agera utefter dessa, och just under den tiden började email kunskaperna snabbt spridas och anammas. Vad passar då inte bättre än en telefon som klarar just dessa uppgifter, vilka var att bland annat ta emot samtal samt kunna besvara email på resande fot.[17] Den första Blackberryn introducerades 1999, och blev en succé i USA.

Det som gjorde den till en succé var att man inte var tvungen och bära runt på en tung bärbar dator, och även då behöva synkronisera email och uppdatera allting på egen hand. Med en Blackberry kunde alla dessa uppgifter skötas per automatik och det underlättade för affärsmännen i deras arbete.

3.3.1 Blackberry OS SDK

Blackberry Widget SDK v1.0 används till för att skapa applikationer till Blackberry.
Hela SDK-paketet består utav; Widget packager, en Blackberry Smartphone simulator, Email simulator, dokumentation och exempelkod. Widgets är småprogram som underlättar för en användare eller används för nöjes skull. Exempel på widgets är väderikon, som återfinns i

[16] http://www.comscore.com/Press_Events/Press_Releases/2010/3/comScore_Reports_January_2010_U.S._Mobile_Subscriber_Market_Share
[17] http://site.ebrary.com.focus.lib.kth.se/lib/kth/docDetail.action?docID=10130499, sid 26

många telefoner och datorer. Blackberry applikationer är små, diskreta och självständiga applikationer som ser ut som vanliga och redan förinstallerade Blackberry applikationer. Applikationerna använder språken; JavaScript, HTML och CSS[18].

3.4 Statistik över användandet av smartphone i USA

Som man kan se och av att döma från bilden nedan (*tabell 1*), är operativsystem från RIM till Blackberry, det system som används mest i USA. Denna statistik som är från en marknadsundersökning är gjord i USA under 2009 och som fortlöpte under tre månader, visar att Apples operativsystem iPhone OS X ligger på andra plats i USA. Detta är ett bra resultat med tanke på att Blackberry telefonerna har funnits i cirka elva år, jämfört med iPhone som har funnits i tre år. Det som är mer intressant är att andelen abonnenter som har införskaffat en telefon med iPhone OS X har ökat med 0.3%, medan den tredje största aktören på marknaden, Microsoft, tappade hela 4 % under samma period. Android från Google har tagit sig in på marknaden och man ser en tendens på att denna aktör bara fortsätter att ta marknadsandelar.

Tabell 1: som visar statistik över de mest vanliga operativsystemen i USA. [19]

Top Smartphone Platforms 3 Month Avg. Ending Jan. 2010 vs. 3 Month Avg. Ending Oct. 2009 Total U.S Age 13+ Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Oct-09	Jan-10	Point Change
Total Smartphone Subscribers	*100.0%*	*100.0%*	*N/A*
RIM	41.3%	43.0%	1.7
Apple	24.8%	25.1%	0.3
Microsoft	19.7%	15.7%	-4.0
Google	2.8%	7.1%	4.3
Palm	7.8%	5.7%	-2.1

[18] http://na.blackberry.com/eng/developers/browserdev/advfeatures.jsp
[19]http://www.comscore.com/Press_Events/Press_Releases/2010/3/comScore_Reports_January_2010_U.S._Mobile_Subscriber_Market_Share

4 OS X SDK

4.1 Historia om Mac OS

För att skapa en översikt av rapporten och förbereda dig som läsare, kommer nu en inledning till Mac OS X och utvecklingen av detta operativsystem.

Redan 1976 släppte man den första datorn med Apples första självständiga och oberoende operativsystem. Det var Steve Jobs, Steve Wozniak, och en tekniker vid namnet Ronald Wayne som var med och lanserade den första datorn gjord av företaget Apple. Denna kom att kallas för Apple 1. Redan något år senare, lanserade man efterföljaren Apple 2.

År 1984 är det året som är mest känt inom Apples kretsar. Det var nämligen det året man lanserade konceptet Macintosh. Det var Steve Jobs som personligen avslöjade nyheten om Macintosh. Det som vi i dagens samhälle är mer vana vid, är just namnet Macintosh. Ett produktnamn på en uppsättning av datorer som kom att släppas under många år, vilket datorerna än idag gör.

Idag är Mac OS X eller även officiellt kallat ”Version 10”, där X:et betyder siffran tio enligt det romerska siffersystemet, det senaste operativsystem från Apple, Denna version av operativsystemet bygger på ett helt nytt koncept med ett nytt filsystem, kodbas, design och hårdvara jämfört med sina föregångare.[20]

Sedan 2001 har man släppt sex stycken ”klient” och ”server” versioner. Den senaste versionen av Mac OS X v10.6 släpptes den 28e augusti 2009. Alla versionsnamn bygger på olika större kattdjur. Den senaste versionen heter ”Snow Leopard”

Apple producerar även mer speciella versioner av Mac OS X till sina fyra andra produkter som används av kunder, såsom iPhone OS, iPod Touch, iPad och en version till Apple TV. Mac OS X bygger mycket på Steve Jobs egna system, som i sig är ett objektorienterat operativsystem. Det systemet heter NeXT[21], som senare kom att kallas OPENSTEP[22], och det är detta system som har lagt grunden för Apples senaste OS. Apple köpte NeXT direkt från Steve Jobs och gavs senare platsen som Chief Executive Offices (CEO) på företaget efter att han valt att komma tillbaka till Apple.

[20] http://www.apple.com/macosx/
[21] http://web.archive.org/web/19970301172356/http://live.apple.com/next/961220.pr.rel.next.html
[22] http://www.kernelthread.com/publications/appleoshistory/7.html

4.2 Utvecklingsverktyg

Apple tillhandahåller sina egna utvecklingsverktyg, och ett av dem är just Xcode.
Detta verktyg ger tillgång till olika kompilerare som karar av C, C++, Objective-C och Java.
Man använder sig av språket Objective-C när man programmerar. Det andra mest förekommande verktyget är Interface Builder (IB).

4.2.1 Verktygen för utvecklig av iPhone och deras historia

I en artikel skriven i början på 2008, beskriver Apple hur utvecklingsverktygen i deras SDK, ska kunna underlätta för programmerare att börja skriva kod för applikationer.[23]
Om vi ser på den helhetsbild som finns idag, i början på 2010, två år efter lanseringen av utvecklingsverktyget ser man direkt vilken påverkan det har gjort för mobilbranschen. Det som har hänt är att många programmerare har fått tillgång till ett nätverk, Apple Development Connection, där man kan ladda ner, diskutera, utbyta information och få idéer om olika programmeringstekniker och efterfrågade applikationer. Ryktet om utvecklingsverktygen har spridit sig och det har blivit alltmer populärt. Fler och fler engagerar sig i denna typ av programmering och man får en insyn i hur man ska tillfredställa de olika behoven för alla applikationer.

Utvecklingsmiljön som används för att skriva program till iPhone är lite annorlunda jämfört med andra liknande smartphones. I exempelvis operativsystemet Android, som utvecklas och distribueras av Google, använder man sig av en javaliknande miljö. Man kan då använda programmet Eclipse för att skriva kod i, vilket många programmerare som skriver i språken C, C++ eller java tycker är ett användarvänligt och smidigt program. Att man även kan använda XML-kod för att skriva program och ändra i Graphical User Interface[24] (GUI), underlättar för de flesta programmerare som till den största delen redan har erfarenhet inom just språket XML.

Android är just nu det snabbast växande mobila operativsystemet, och det är även den plattform som konkurerar med iPhone OS X.[25] Därför ger det programmerarna som skriver i Apples SDK en möjlighet att utvecklas eller viljan och motivationen att skriva bättre och snabbare program. Detta kanske rentav sätter press på Apples utvecklare att hitta nya och enklare metoder för att underlätta för sina, för varje dag växande i antal, programmerare.

4.2.2 Utvecklingsmiljö

Miljön som iPhone-utvecklare använder sig av, skriver kod i och som det mesta är implementerat i kallas på engelska för ”Objective-C Programming Language”.[26]
Denna typ av programmeringsspråk är ett enkelt och sofistikerat språk för objektorienterad programmering.[27] Språket fungerar som en utbyggnad och förlängning av språket C.

[23] http://www.datormagazin.se/nyheter/article213804.ece
[24] http://www.pcw.co.uk/personal-computer-world/features/2045763/men-really-invented-gui
[25] http://www.idg.se/2.1085/1.288649/android-snabbast-vaxande-mobiloperativet
[26]http://developer.apple.com/Mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html

Denna typ av utveckling har skett för att man enkelt ska kunna utveckla och skriva objektorienterat i språket C och ge språket full kapacitet för att skriva som man vill, där just funktionen med objektorienterat språk ej tidigare funnits tillgängligt.

Det som identifierar den utvecklingsmiljö som vanligtvis kännetecknar objektorienterad programmering är främst:

- *Ett objektorienterat programmeringsspråk, exempelvis C/C++*
- *Ett bibliotek av objekt*
- *En uppsättning av utvecklingsverktyg, exempelvis Xcode och Interface Builder*
- *En kompilator*

Det som är själva kärnan i programmeringen, rent generellt, är den första komponenten, vilket är programmeringsspråket. I detta fall används Objective-C med inriktning på C/C++.
Den andra komponenten är Cocoa, Mac OS X Objective-C application frameworks.
De två vanligaste utvecklingsverktygen som används är Xcode och Interface Builder (IB).[28]
Redan här skiljer sig programmeringen till skillnad från Android, t.ex. med IB, medan man i Android kan använda XML-kod använder man i iPhone OS, Interface Builder.

De Apple kompilatorer som finns idag är baserade på ”GNU Compiler Collection”[29].
Då C/C++ kod är väldigt likt Objektiv-C syntax är även kompilerarna beredda att exekvera de tre, inte alltför olika, språken och deras källkod. Kompilatorn känner igen de olika filerna, mer specifikt att den känner igen Objektiv-C syntax och källkoderna, genom filändelsen ”.m”, vilket är väldigt likt hur en kompilator känner igen en fil med ändelsen ”.c”. Denna typ av filändelse representerar en källkod skriven i språket C och den syntaxen. Samma sak gäller för kompilatorn när den känner igen en fil skriven med C++ syntax, då har den filändelsen ”.mm”.

4.2.3 Objective-C

Ett objekt associerar data med den specifika operationen som påverkar eller kan använda den data. I Objective-C kallas dessa operationer *objektets metoder* och den data de påverkar kallas för instans variabler. I princip kan man säga att ett objekt packar ihop en datastruktur (instans variabler) och en grupp av metoder till en självständig programmeringsenhet.

I ett Objective-C program måste man hela tiden deallokera objekt då de inte används, det för att programmet inte ska ta onödigt mycket minne. Men det är viktigt att man inte deallokerar objekt som för tillfället används.

I Objective-C[30] erbjuds man två olika miljöer som klarar av minneshantering:

- *Referens beräkning*; där man själv ansvarar för livslängden för de objekt som används.

---

[27]http://developer.apple.com/Mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html
[28] http://developer.apple.com/technologies/tools/xcode.html
[29] http://developer.apple.com/mac/library/documentation/DeveloperTools/gcc-4.0.1/gcc/Standards.html
[30]http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html

- *Skräphantering*; där man överlåter ansvaret för minneshanteringen och livslängden för objekten till en automatisk ”samlare”.

För att få ett objekt att göra som man vill, måste ett meddelande skickas, vilket i sin tur berättar att den ska anropa/applicera en metod. Dessa meddelanden är avgränsade med hakparenteser i språket Objective-C.

Ex1.
Exempel på hur det ovannämnda stycket kan skrivas i Objective-C. [31]

```
[receiver message]
```

”*Receiver*” är ett objekt och ”*message*” säger åt den vad den ska göra. I källkoden är meddelandet helt enkelt namnet på en metod och vilka argument som skickas till den.

Ex2.
Ett annat exempel på vad som sker. I koden nedan så säger meddelandet åt objektet *”myRectangle”* att exekvera sin display metod, vilket betyder att den kommer att rita/synliggöra rektangeln.

```
[myRectangle display];
```

Ex3.
Ett annat exempel på multipla argument visas nedan. Med multipla argument menas att en metod kan ta olika indata som den sedan ska utföra. Här kommer ursprungskoordinaterna att ändras till de värden man har satt. Man ser redan i koden på X- och Y-axeln vilka koordinater som ändras.

```
[myRectangle setOriginX: 30.0 y: 50.0];
```

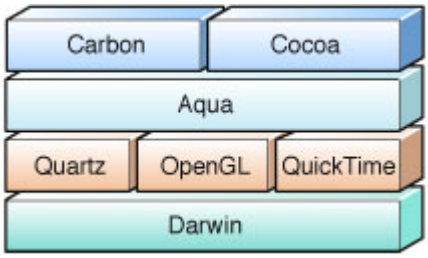
4.2.4 Cocoa

För en programmerare som använder Cocoa[32] för första gången, ser man en uppsjö av stora och outforskade tekniska möjligheter. Hela konceptet med designen, verktygen, gränssnittet och även programmeringsspråket kan kännas främmande i början. Med hjälp av *”Cocoa Fundamentals Guide”* får man en bra inledning och beskrivning av programmet och hur man snabbt kommer igång med sitt arbete.

Utvecklingsverktyget Cocoa, som annars används i Mac-datorerna, har nu tillämpats för att kunna användas i iPhone OS. Man har nu istället fokuserat på själva touchfunktionen och optimeringen av denna. User Interface Kit (UIKit) tillhandahåller de basverktyg man behöver för att kunna utveckla och implementera grafiska applikationer för iPhone OS. Som tidigare nämnt, så bygger UIKit på samma infrastruktur och framework som man hittar på Mac OS X. Med detta utvecklingsverktyg får man tillgång till speciella GUI kontroller, knappar och även ett fullskärmsläge i iPhone OS. Man får även tillgång till hantering av program med accelerometern och multi-touch gester i det läget.

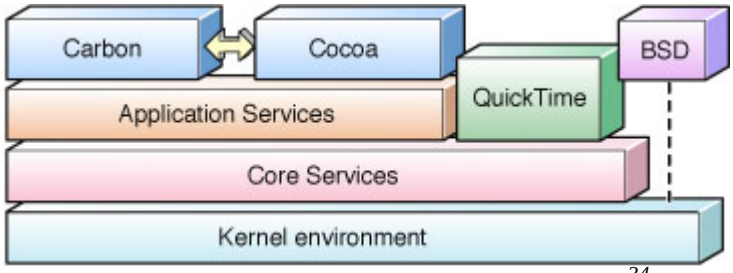
[31] http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Articles/ocDefiningClasses.html#//apple_ref/doc/uid/TP30001163-CH12-SW1
[32] http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/CocoaFundamentals/Introduction/Introduction.html

Det som är mest relevant för denna rapport är hur Cocoa och dess minneshantering fungerar i en iPhone men för att kunna ge en bra översikt över programmet måste vi förklara kort hur det fungerar i Mac OS X för att senare koppla samman det med iPhone OS.

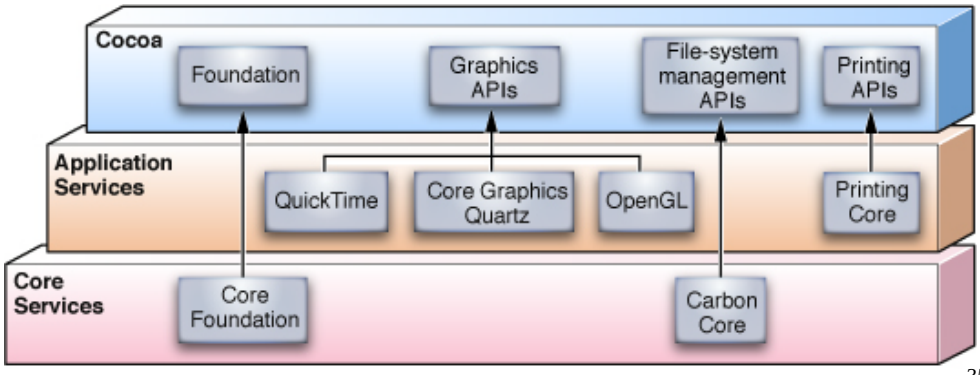


Figur 1: Mac OS X arkitektur–förenklat perspektiv.[33]
http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html, Besökt 2010-04-03

Figuren ovan är bra för den som ej tidigare har programmerat eller är ovan vid Mac OS X och dess komponenter och beroenden. Den visar med enkelhet hur Cocoa är integrerat i Mac OS X operativsystem.



Figur 2: Cocoa i arkitekturen i Mac OS X.[34]
http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html, Besökt 2010-04-03



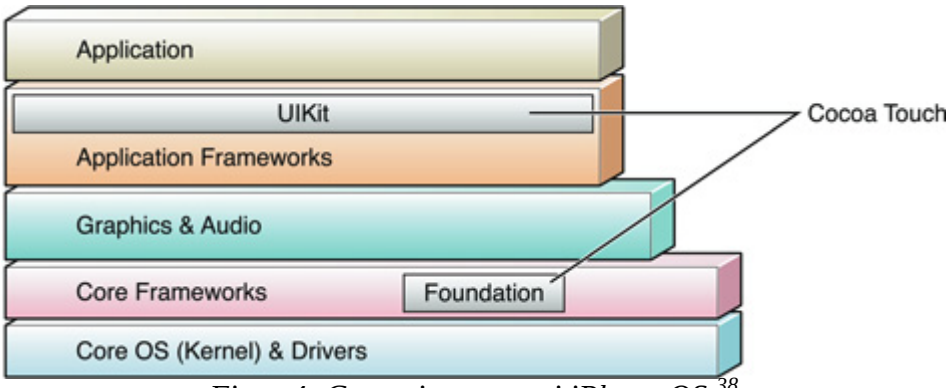
Figur 3: Mer detaljerad bild av Cocoa arkitekturen och mer om beroenden.[35]

[33] http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html
[34] http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html
[35] http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html, Besökt 2010-04-03

Här ovan beskrivs arkitekturen mer detaljerat och av att döma utifrån diagrammet, är Mac OS X uppbyggt som lager av mjukvara som förgrenar sig ut till de olika applikationsmiljöerna. Man ser även hur de olika lagren är beroende utav varandra och hur dem interagerar med varandra. Det mesta av Cocoa är implementerat i Objective-C, vilket är kompilerat så att det kan köras i snabba hastigheter, men som samtidigt är väldigt dynamiskt och flexibelt. Då Objective-C programmering bygger på språken C/C++ kan man använda en kombination av båda språken i Cocoa Touch applikationerna[36].

Det som utmärker Cocoa är att under tiden applikationen körs, kommer Objective-C körningen att exemplifiera objekt så de kan exekveras logiskt, och inte bara på hur dem har definierats vid kompilering. Exempelvis, kan en redan exekverad Objective-C applikation ladda in ett gränssnitt (en nib fil skapad av Interface Builder)[37], ansluta till Cocoa objekt i gränssnittet av applikationens kod, och sedan körs den rätta metoden i koden när man väl har tryckt på User Interface (UI) knappen. Därmed behövs ingen andra exekvering av koden.



Figur 4: Cocoa integrerat i iPhone OS.[38]

http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html, Besökt 2010-04-03

Som ett komplement till UIKit, har Cocoa Touch kollektionen av frameworks, (ramverk/strukturer) många inkluderade verktyg och tillbehör. Alla dessa verktyg är nödvändiga för att skapa förstklassiga iPhone applikationer, allt från 3D grafik till ljud och även programåtkomst och tillgång till GPS koordinater. Några exempel på dessa frameworks är ”Core Animation”, ”Core Audio” och ”Core Data”.

Utifrån en jämförelse av figur 2 och figur 4 går det att urskilja de skillnader som finns mellan de olika operativsystemen, Mac OS X och iPhone OS, fastän de är väldigt få. Som i Mac OS X består även iPhone OS av mellanlager som består utav Core services ramverk (kärnkomponenter), ljud- och grafikramverk och bibliotek. Precis som i *figur 2* har även de olika lagren beroenden på varandra, och det ena lagret har ett beroende på underliggande lagret.

[36]http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html
[37]http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html
[38]http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

De bibliotek och ramverk till iPhone OS och som stödjer UIKit, är delmängder av de ramverk och bibliotek som finns i Mac OS X. Det finns specifika ramverk, både privata och publika, som utmärker iPhone OS och dessa är med början från det nedersta lagret:

Core OS
Här ingår kärnan, filsystemet, nätverksstruktur, säkerhet och ström- och elhantering. Man finner även ett libSystem bibliotek och APIs för flertalet tjänster.

Core Services
Som det låter på namnet av detta ramverk, så ingår kärntjänster, såsom sträng förändring, nätverk, kontakthantering och URL hjälpprogram.

Media
Ramverken här är beroende av Core Services lagret och tillhandahåller både grafiska och multimedia tjänster till Cocoa Touch lager. Dessa tjänster är Core Graphics, OpenGL ES (Embedded Systems), Core Animation, Core Audio och videouppspelning.

Cocoa Touch
I detta lager har ramverken direkt stöd för de applikationer baserade för iPhone OS. Dessa ramverk inkluderar två Objective-C ramverk som är väldigt viktiga för utvecklingen av applikationer för iPhone OS: [39]

- Det första är UIKit ramverket som tillhandahåller de objekt en applikation gör synlig i gränssnittet och ramverket definierar även strukturen för beteendet för en applikation, där även ”event handling” och uppritning är inkluderade.

- Det andra som kallas *”The Foundation framework”* definierar de mest nödvändiga och grundläggande beteenden av objekt, bildar mekanismer för deras hantering, förser objekt med primitiva datatyper och tjänster för operativsystemet. Foundation är ett objektorienterat skal till Core Foundation ramverk.

4.2.5 Översikt av utvecklingsflöden

Utveckling av applikationer skiljer sig en hel del från när man skriver för Mac OS X och till att skriva för iPhone OS. Det är inte bara verktygen som används som skiljer sig åt, utan hela utvecklingsflödet skiljer sig.

Utveckling för iPhone OS är lite mer komplext då arbetsflödet skiljer sig. Innan man kan börja utveckla för iPhone OS måste man först och främst registrera sig som utvecklare för den specifika plattformen, för att sedan kunna utveckla och lansera en applikation. Men det finns några steg som man måste följa då man utvecklar för iPhone OS.
I tabellen på följande sida ser man skillnaderna och likheterna mellan de olika utvecklingsmiljöerna.

[39] http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

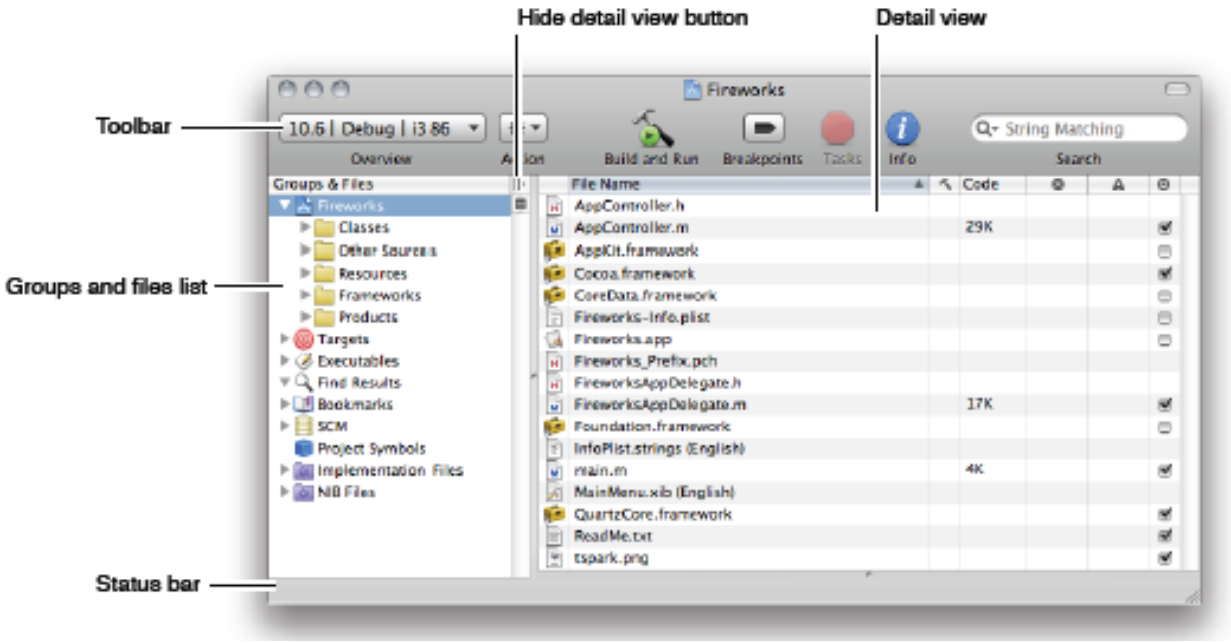
Tabell 2: En tabell som beskriver skillnader i arbetsflödet mellan Mac OS X och iPhone OS.[40]
http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

	Mac OS X	**iPhone OS**
1	i Xcode skapar man ett projekt genom att använda en färdig mall från Mac OS X SDK	Konfigurera enheten. Detta resulterar i att man måste installera och inneha de obligatoriska verktygen, ramverken och andra komponenter. I Xcode skapar man ett projekt genom att använda en färdig mall från iPhone OS SDK
2	Skriv kod och genom att använda IB konstruera egna gränssnitt för applikationer	Man skapar applikationer lokalt.
3	Testa och felsöka applikationer genom att använda Xcode felsöknings möjligheter.	Testa och felsöka applikationer genom att använda antingen iPhone OS simulator eller fjärrstyrt i enheten. Och sedan sker mätning av prestanda på samma sätt som i Mac OS X.
4	Mäta en applikations prestanda genom att använda en eller flera prestanda och tweaking verktyg.	Använder och felsöker med iPhone OS simulatorn

[40]http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

4.3 Hur gör man för att utveckla i Xcode?

Det mesta arbetet när man jobbar i Xcode sker i projektfönstret. Det man ser i projektfönstret är hur filer, målfiler och exekverbara filer organiseras och visas. Detta underlättar och reducerar jobbet avsevärt för programmerare och för att kunna utnyttja programmet fullt ut ska man ha god kännedom av just projektfönstret.



Figur 5: Visar hur projektfönstret ser ut i Xcode.[41]
http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/A_Tour_of_Xcode/010Xcode_Features_Overview/FeaturesTake2.html#//apple_ref/doc/uid/TP30000890-CH220-SW3, Besökt 2010-04-05

Groups & Files list
Tillhandahåller en översikt av projektets innehåll. Här kan man flytta filer och mappar och organisera för optimerad arbets- och utvecklingsmiljö.

Detail button
Man kan även få en detaljerad vy av det man arbetar med, och denna knapp kan man sätta på/stänga av. Där kan man utforska filerna och elementen man arbetar med och se en detaljerad beskrivning av dessa.

Toolbar
I själva verktygsraden kan man få snabb tillgång till de flesta Xcode kommandon.

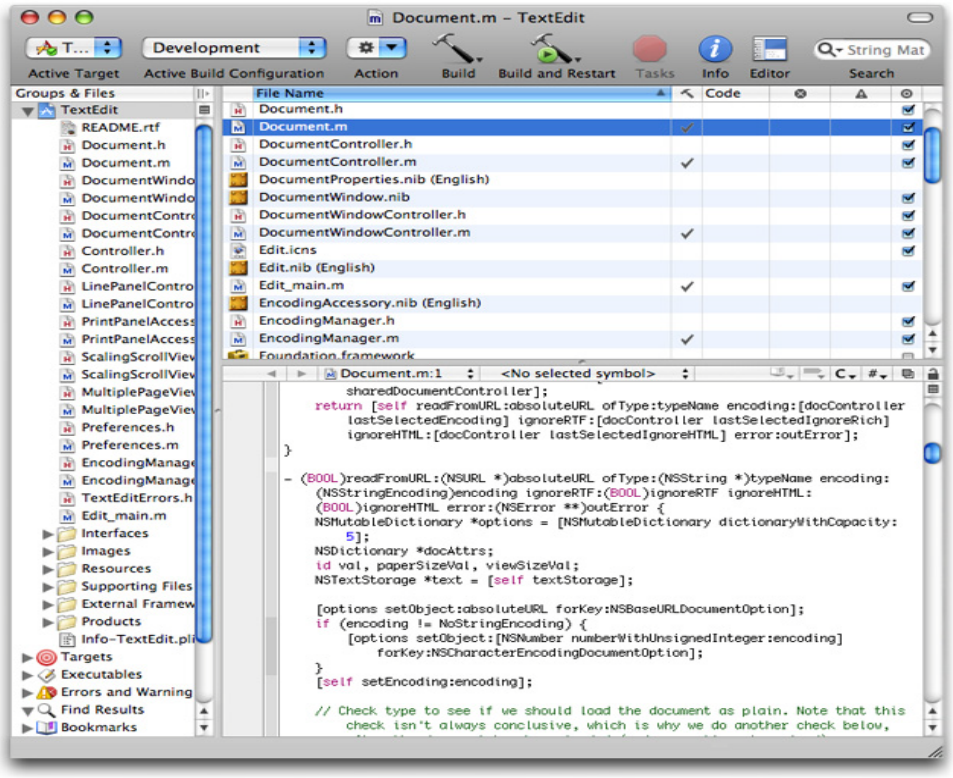
[41] http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/A_Tour_of_Xcode/010Xcode_Features_Overview/FeaturesTake2.html#//apple_ref/doc/uid/TP30000890-CH220-SW3

Status bar
I statusraden får man ett meddelande för specifikt projekt man arbetar med. Under en körning, som exempelvis en indexering, kommer Xcode att visa en utvecklingsindikator i statusraden för att visa aktuell status för uppgift.

Favorites bar
Som det låter på namnet, kan man lagra ofta förekommande platser i projektet, för att snabbt komma åt dessa.

Vad Xcode egentligen gör är att det är den enhet som förser Apples integrerade utvecklingsmiljö, Integrated Development Environment (IDE) för Mac OS X och iPhone OS. Det är samtidigt en applikation som tidigare beskrivet tar hand om det mesta kring ett projekt. Xcode är även integrerat och har ett väldigt bra samspel med IB.

Med hjälp av Xcode som tidigare nämnt, bygger man upp sina projekt från källkod som är skriven i C, C++, Objective-C och Objective-C++. I Mac OS X genereras exekverbara filer av alla typer där ramverk, plugin och applikationer ingår. Medan det för iPhone OS endast är möjligt med exekverbara applikationer, inget annat.



Figur 6: Ett exempel på ett projekt i Xcode.[42]
http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/A_Tourof_Xcode/010Xcode_Features_Overview/FeaturesTake2.html#//apple_ref/doc/uid/TP30000890-CH220-SW3, Besökt 2010-04-05

[42] http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/A_Tourof_Xcode/010Xcode_Features_Overview/FeaturesTake2.html#//apple_ref/doc/uid/TP30000890-CH220-SW3, Besökt 2010-04-05

4.3.1 Interface Builder

Det andra stora utvecklingsverktyget för Cocoa projekt är Interface Builder. Detta är precis som namnet redan avslöjar, ett grafiskt verktyg för skapandet av användargränssnitt. Detta verktyg har varit med sedan början av Cocoa och gick då under namnet NeXTSTEP. Man kan även använda detta för Mac OS X när man ska skapa applikationer just för det operativsystemet.

4.3.2 iPhone OS simulator

När en programmerare/utvecklare skapar iPhone OS projekt kan man välja iPhone OS simulatorn som plattforms SDK för specifikt projekt.[43] Simulatorn kommer automatiskt att köras via anrop från Xcode när man kompilerar och exekverar projektet, vilket i sin tur presenterar applikationen precis som om den skulle köras i telefonen. Då kan man även ändra delar av användargränssnittet. Man kan även använda simulatorn som verktyg för att felsöka och gå igenom applikationen innan man för över och implementerar den i telefonen.

Som med de flesta simulatorer kan man inte få en alldeles perfekt och realistisk körning, men man får en bra bild av hur applikationen kommer att köras. Ett exempel på att simulatorn inte riktigt kan testa allt är att man bland annat måste använda musen istället för fingrarna samt att touchfunktionen skiljer sig när man arbetar på en dator. Simulatorn använder även olika ramverk och delar av ”*Foundation*” när den simulerar och inte iPhone OS egna ramverk. Därmed kan man förlora en hel del i testningen och man kanske inte får korrekta resultat. Det man ska ha i åtanke är att man kör denna simulator på en dator som har mer minne och kraft än en iPhone, vilket är väldigt viktigt att tänka på när man programmerar applikationer.

4.3.3 Garbage collector i Mac OS X

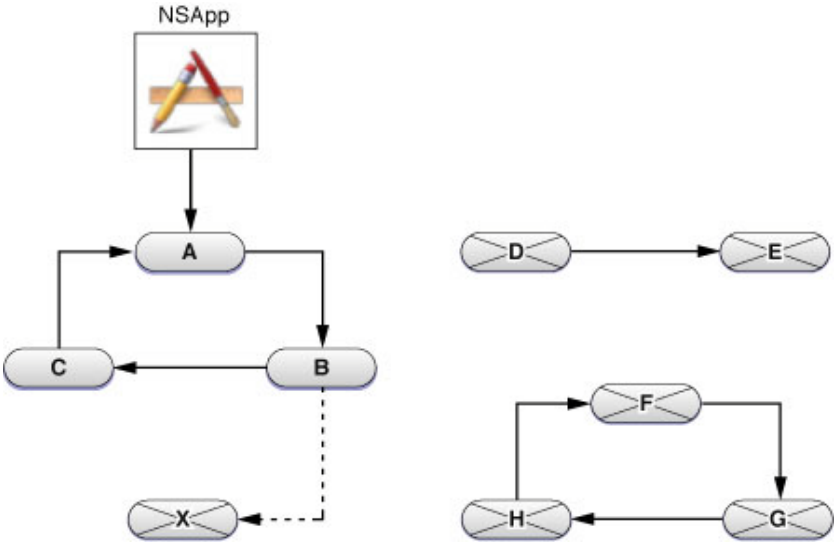
När man använder Cocoas skräphantering[44], tar den hand om minnet i applikationen för programmeraren. Detta innebär att alla Cocoa objekt är med och tas hand om av skräphanteringen. Det vill säga att man inte behöver hålla reda på om ett objekts ”*retain*” beräkning är avslutad, med andra ord att ett objekt inte är ”*live*” eller att det minnet det har använt blir återlämnat så andra processer kan använda det, när objektet är klart med det.

Skärphanterarens mål är att skapa en mängd av objekt som är nåbara för att sedan validera dessa med de objekt som används i applikationen och förkasta resten som inte används. När en hanterare startar, kommer den att initiera mängden med de kända rotobjekten, och sedan följer hanteraren ”*starka*” referenser mellan objekt för att slutligen lägga till dessa i mängden. För att sedan avgöra vad som är skräp och vad som inte är det, följer hanteraren en kedja av alla ”*starka*” referenser och selekterar då bort allt som inte tillhör kedjan. När detta är identifierat och skräpet tagits hand om frigörs omedelbart det minnet de objekten tidigare har tagit upp.

[43] http://developer.apple.com/iphone/library/documentation/Xcode/Conceptual/iphone_development/125-Using_iPhone_Simulator/iphone_simulator_application.html
[44]http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html

Som tidigare nämnt, så anses bara det som inte är rotobjekt och som inte har ”*starka*” referenser som skräp. Mängden av rotobjekt inkluderar bland annat globala variabler, stack variabler och objekt med externa referenser, vilka alla inte anses som skräp. Genom att identifiera anropsstacken av varje Cocoa tråd som körs, hittar man vilka rotobjekt som är nåbara och deras referenser, vilka då i sin tur ingår i rotmängden.

Det finns sammanfattningsvis två typer av referenser, ”*svaga*” och ”*starka*”, där den senare är synlig för hanteraren medan den förstnämnda inte är det. Detta illustreras och beskrivs i figuren nedan.



Figur 7: ”Starka” och ”svaga” referenser mellan objekt.[45]
http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1, Besökt 2010-04-05

Man kan i den vänstra bilden se att A, B och C har ”*starka*” referenser till roten, medan X har en ”*svag*” referens, vilket leder till att skräphanteraren tar hand om detta.

Medan i den högra bilden, kommer D och E anses som skräp av hanteraren fastän de har ”*starka*” referenser, men de saknar referens till ett rotobjekt.

Skräphanteraren är ett valfritt alternativ och tillägg som man som programmerare själv kan aktivera genom att ”*flagga*” delar av koden för att kompileraren ska känna igen var i koden och från vilken punkt skräphanteraren ska initieras.

Det finns tre möjliga skräphanterar alternativ[46]:

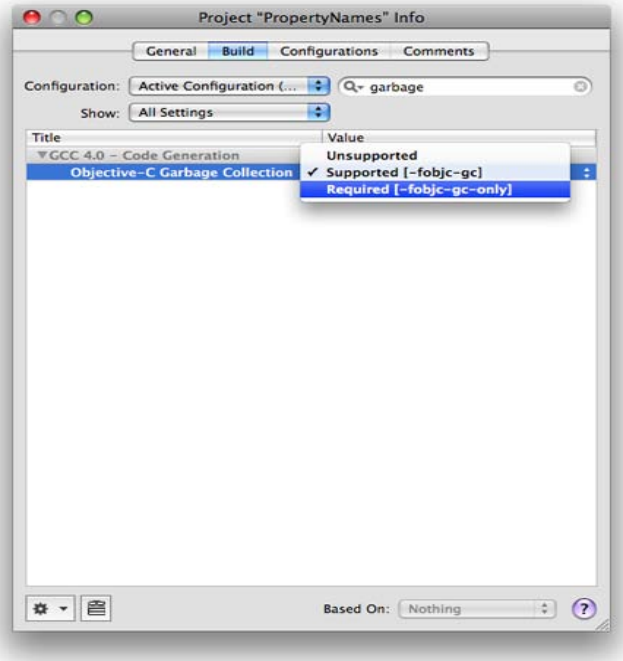
- Ingen flagga, därmed ingen skräphanterare.
- `-fobjc-gc-only` Betyder att den logiska skräphanteraren är igång.

[45] http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1, Besökt 2010-04-05
[46] http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1

En kod som är kompilerad där skräphanteraren varit ett krav, är sagt att inte behöva använda Cocoas ”*retain/release*” metoder och kanske inte implementeras i någon applikation som inte använder sig utav en skräphanterare.

- `-fobjc-gc` Betyder att både skräphanteraren och ”*retain/release*” är aktiva.

  En kod som är kompilerad med skräphanteraren som stöd är förutsagd att den innehåller ”*retain/release*” metoder och kan därmed implementeras in en applikation.



Figur 8: I Xcode valbara alternativ för skräphanteraren.[47]
http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1, Besökt 2010-04-05

Viktigt att tänka på samt att försöka undvika under programmering då man vill optimera minnet och inte slösa på de få resurser man redan har.

4.3.4 Begränsningar eller möjligheter till att utvecklas?

Dagens mobiltelefoner har utvecklats mycket på senare år, och man ser en uppåtgående trend. Men fortfarande har man en hel del begräsningar som man måste tänka på när man programmerar applikationer. I tabellen nedan ser man vilka skillnaderna är mellan en dator och en telefon, och just specifikt i detta fall en iPhone 3GS. Detta är för att man som läsare ska bli mer insatt i det som senare ska förklaras i rapporten om just iPhone och begränsningar kring minneshanteringen och allokeringen. Ska tabellen nedan endast ses som en begränsning eller en möjlighet till att utveckla dagens hårdvara i enheterna?

[47] http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1

Tabell 3: Förklarar skillnaderna mellan en dator och iPhone 3GS

	Dator	**iPhone 3GS** [48]
Minnet(Lagringsminnet)	Hårddiskar, runt 500 GB	Inbyggt 16/32 GB
RAM	4 GB	256 mb
Processor	3000 MHz (flera kärnor)	600MHz (en kärna)
Strömförsörjning	Elnät, ca 35A	Batteri, ca 400mA

Arkitekturen mellan en stationär dator såsom en iMac och en iPhone skiljer sig åt, och detta sätter sina spår då man ska börja programmera. Som utvecklare finns det mycket att tänka på, exempelvis hur skräphanteraren och hur minneshanteringen fungerar i en telefon. Vi kommer mer grundligt försöka titta på exempelkod för att se vad som kan gå fel och hur det påverkar minnet och skräphanteringen.

Med dessa specifikationer kommer vi att försöka besvara några av de begränsningar som uppstår och vad man ska tänka på när man programmerar en applikation för Mac OS X och främst för iPhone OS. Nedan följer mer information om just iPhone och dess innehåll.

5 iPhone

5.1 Prestanda och grafik

Att grafiken har bra kvalitet är en viktig del för ett bra och snyggt användargränssnitt. Högkvalitativ grafik gör inte bara att en applikation ser bra ut, utan det gör också att en applikation ser ut som en naturlig förlängning av resten av operativsystemet. iPhone OS ger två huvudsakliga möjligheter för att skapa högkvalitativ grafik i ditt system: OpenGL eller naturlig rendering med Quartz, Core Animation och UIKit.

OpenGL ramar är inriktade främst mot spelutveckling eller tillämpningar som kräver hög bildhastighet. OpenGL är ett C-baserat gränssnitt som används för att skapa 2D- och 3D-innehåll på stationära datorer. iPhone OS stödjer OpenGL ritningar via OpenGL ES, Embedded System, som ger stöd för både OpenGL ES 2.0 och OpenGL ES v1.1 specifikationer.[49] OpenGL ES är speciellt utformad för att användas i inbyggda hårdvarusystem, vilket skiljer den på många sätt från versioner som annars finns på skrivbordet och från OpenGL.

Open GL ES, hade mycket funktionalitet borttagen från den ursprungliga OpenGL. Två av de mer betydande skillnaderna mellan OpenGL ES och OpenGL är avlägsnandet av *glBegin* och *glEnd*, vilket gör att den kräver semantik för primitiv rendering (till förmån för vertex arrayer) och införandet av fasta datatyper för vertex-koordinater. Många andra områden av funktionalitet har tagits bort i version 1.0 för att producera ett lätt gränssnitt, till exempel, quad, texgen, linje och punkterade polygon.[50]

För utvecklare som vill ha en mer objektorienterad strategi ger Quartz, Core Animation och

[48] http://www.gsmarena.com/apple_iphone_3gs-2826.php
[49] http://www.opengl.org/sdk
[50] http://www.khronos.org/opengles/1_X

iPhone OS grafiken stöd i UIKit. Quartz är viktigast för ritning av gränssnitt, stöd för vägbaserade teckning, kantutjämnade rendering, toning av mönster, bilder, färger, samordna space-transformationer, och PDF dokument. Core Animation ger underliggande stöd för att animera förändringar i UIKit.

I iPhone OS är, all ritning oavsett om det handlar om OpenGL, Quartz, UIKit eller Core Animation, skapad inom ramen för en UIView objekt. Visningar definieras av den del av skärmen där teckning sker och är synlig.

5.2 Begränsningar vid programmering av programvara för iPhone

iPhone OS är ett operativsystem för mobila enheter. Precis som på alla mobila enheter, är CPU-prestanda och RAM-minnen begränsade jämfört med stationära datorer. Det betyder att när man skriver mjukvara för iPhone, är optimerad kod avgörande för en bra och smidig användarupplevelse.

Den totala mängden tillgängligt minne för ett program som körs på iPhone OS är begränsat. Av denna anledning är det viktigt att programvaran hanterar minne noga och inte läcker resurser.

Batteritiden för en enhet som kör iPhone OS är direkt relaterade till Central Processing Unit, (CPU), Graphics Processing Unit (GPU), och utnyttjandet av telenäten. Så det är också viktigt att inte utnyttja CPU och GPU på onödigt arbete och undvika att hålla nätverksmaskinvaran, såsom e-post funktion, aktiva i onödan.

5.2.1 Batteri

iPhone har ett internt uppladdningsbart batteri. Batteriet är ungefär som andra mobiltelefoners batterier, men den största skillnaden är att ett iPhone batteri inte kan bytas ut på egen hand. För att man skall kunna byta sitt batteri krävs en licensierad återförsäljare och det kan bli en dyr historia. Då batteriet är ett problem eftersom det förr eller senare urladdas, måste en programmerare tänka på programmen och deras elförbrukning. Det har inte bara med att batteriet urholkas, utan även för att om många tunga applikationer körs samtidigt blir det en dålig batteriprestanda.

När man programmerar för iPhone OS kan man läsa på ”Apples iPhone application Programming guide” att man ska försöka stänga av så många onödiga processer så fort som möjligt.[51]

Processer som fungerar som en batterislukare är bland annat:

- *Lokaliseringsservice* – Applikationer som använder sig av funktionen för att lokalisera var man befinner sig drar mycket ström. Både GPS och kartfunktion måste vara aktiva.
- *Push-meddelande* – Push funktionen sänder en varning till användaren om något ändras eller inkommer till programmet, exempelvis ett email. För att pushfunktionen

[51] http://developer.apple.com/iphone/library/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AdvancedFeatures/AdvancedFeatures.html

skall vara till nytta behöver den kolla mot förändringar i programmet under ett intervall, exempelvis varje minut. Det resulterar i sämre batteritid för telefonen vilket aktiverar onödiga processer som drar mer ström.

- *Hämtning av data* – Vid kontinuerlig hämtning från externa källor krävs oftast flera steg och involvering av batterikrävande processer. Vid kontinuerlig hämtning rekommenderas att man minskar ner till mindre frekvent uppdatering, såsom en gång i minuten istället för varje sekund.
- *Bluetooth* – Om Bluetooth behövs inom någon applikation skall den genast stängas av direkt efter användning för att minimera onödig batterikonsumtion.
- *EQ* – Då musik spelas som bakgrundsmusik i din applikation kan man erbjuda equalizer för att justera volym samt balans, detta är något man gärna bör stänga av, då en sådan process förbrukar mycket batteri. Det rekommenderas att bara ha en on/off knapp för volymen.

Ett vanligt iPhone 3GS batteri ger upp till 5 timmars taltid på 3G, 12 timmars taltid på 2G, 5 timmars Internetanvändning på 3G, 9 timmars Internetanvändning på Wi-Fi, 10 timmars videouppspelning eller 30 timmar av ljuduppspelning på en full laddning av ursprungliga kapaciteten. Dessutom fungerar iPhone upp till 300 timmar i standby-läge.[52]

5.2.2 Minne

I alla program som skrivs, måste man se till att man kan hantera resurser effektivt. En sådan resurs är ett programminne. I ett Objective-C program måste man se till att det man har skapat i minnet avregistreras för att inte det skall ligga kvar och ta upp onödigt utrymme.
Vid ett mer komplext system kan det vara svårt att avgöra exakt när objektet inte längre behövs. För att man ska hålla koll på när och var man ska avregistrera minnesplatser finns det ett antal regler.

Följande är den grundläggande regeln:

- Man blir ägare till ett objekt om man skapar den med hjälp av en metod vars namn börjar med "*Alloc*" eller "*new*" eller innehåller "*copy*" (exempelvis *Alloc, newObject eller mutableCopy*) eller om man sänder ett ”*retain*” meddelande. Programmeraren är ansvarig för avsägandet av äganderätten av objekt man äger, genom att ange ”*release*” eller ”*autorelease*”. Vid alla andra tillfällen man får ett objekt måste man vara noga med att avsäga sig ägandeskapet och överlåta objektet till skräpsamlaren.

Följande regler härstammar från den grundläggande regeln ovan:

- Som en följd av den grundläggande regeln, är att om man behöver lagra ett mottaget objekt som en egenskap av en instansvariabel, måste det behållas eller kopieras.

- Mottagna objekt är normalt garanterat att förbli giltiga i den metod som objekten togs emot i (undantag vid flertrådade program och några distribuerade objektsituationer). Denna metod kan också skicka tillbaka objektet till dess anropare.

[52] http://www.apple.com/batteries/iphone.html

- Använd ”*retain*” i kombination med ”*release*” eller ”*autorelease*” då det behövs för att förhindra att ett objekt ogiltigförklaras.

5.2.3 iPhone OS Operativsystem

Det största problemet som programmerare stöter på när de utvecklar för iPhone är Apples restriktioner. Programmeringsmässigt kan man dessa problem med tre punkter:

- *En applikation i taget* – Att endast en applikation kan köras samtidigt ger dig problem vid t.ex. utvecklandet av internetapplikationer. Om man som användare surfar och det ringer kommer man kopplas ifrån internetuppkopplingen. Om man skulle befinna sig på en säker anslutning innebär det att användaren måste logga in igen med ett användarnamn och lösenord, vilket är ett irriteringsmoment från användarperspektiv.

- *Ingen tredjeparts applikation i bakgrunden* – Att inga tredjeparts applikationer kan köras i bakgrunden innebär problem vid kontakt mot exempelvis en server. Om man i sin programvara är uppkopplad mot en server, exempelvis en chattklient, och man får ett inkommande samtal kommer uppkopplingen inte kunna ligga latent i bakgrunden. Detta medför att man inte kommer kunna ta emot meddelanden samtidigt som man är upptagen i ett samtal. Användaren kommer att avbryta uppkopplingen och chattklienten kommer visa att man är offline.

- *Ingen support för äldre OS-versioner* – Att OS 3.0 inte stödjer äldre versioner av iPhone OS blir ett problem då man som programmerare inte vill stänga ute äldre användare. Detta går i dagsläget tyvärr inte att lösa.

5.2.4 Processor

Systemet tilldelas CPU-tid, som man blir tvungen att använda på bästa möjliga sätt. Eftersom både Mac OS X och iPhone OS genomför symmetrisk ”*multiprocessing*”, vilket betyder att man använder flera trådar för att köra flera processer, ges varje tråd i systemet en liten bit av den tilldelade tiden, högst tio millisekunder för att kunna köras.
I slutet av denna process, eller i många fall före, gör systemet så att den tar tillbaka kontrollen över processen och ger den till en annan tråd.

I ett vanligt system med många aktiva trådar, där varje som tråd används blir tilldelad full CPU-tid, skulle resultaten bli fruktansvärda[53]. Detta leder till att ett av de viktigaste målen för utveckling och programmering av applikationer är: ”*Om ditt program inte har något att göra, bör det inte konsumera CPU-tid*”. Det mest optimala tillvägagångssättet för att uppnå detta är att använda sig av en händelsebaserad modell. Med hjälp av modern händelsehantering, såsom de som finns inbyggda i Cocoa och iPhone OS, kommer endast trådar att köras när det finns arbete att utföra.[54]

[53]http://developer.apple.com/iphone/library/documentation/Performance/Conceptual/PerformanceOverview/BasicTips/BasicTips.html#//apple_ref/doc/uid/TP40001410-CH204-BBCIFICC
[54]http://developer.apple.com/iphone/library/documentation/Performance/Conceptual/PerformanceOverview/BasicTips/BasicTips.html#//apple_ref/doc/uid/TP40001410-CH204-BBCIFICC

När en process ska göra något, bör det använda CPU-tid så effektivt som möjligt.
Detta innebär bland annat att välja algoritmer som är lämpliga för den mängd data den förväntas hantera. Det betyder också att använda andra systemresurser, som en tillgänglig vektorenhet eller en grafikprocessor, för att utföra specifika aktiviteter.

Det leder till nästa mål:
Flytta arbete ur CPU så fort som det möjligtvis går.

Då en beräkning eller en process är utförd i processorn skall man i största möjliga mån, flytta processen till andra delar av telefonen som exempelvis minnet. Detta för att man ska försöka minska att CPU:n inte blir överbelastad.

5.3 Utvecklingsmöjligheter

Ingen vet vad som händer inom Apple. Få företag har sådan hård tystnadsplikt, vilket leder till att det är ytterst sällan rykten sprids i press om framtida produkter och lanseringar.
Det man kan anta är i görningen idag, i början på 2010, är att Apple har sökt patent på en teknik som kallas för ID-app.[55] Denna applikation använder sig utav kameran och GPS:en för att lokalisera och hitta information om föremål som man tar kort på. Till exempel kan man ta en bild på Eiffeltornet, vilket leder till att applikationen ger dig information från exempelvis wikipedia, en hemsida med källor och förklaringar till diverse ämnen.

Ett annat patent man har sökt fokuserar på ansiktsigenkänning, som också kan användas för en mängd olika ändamål. Det kan ge dig information om en person bara genom att ta en bild på personen med kameran för att sedan få information om personen ifråga, eller så kan det användas för en rad olika säkerhetstjänster, så att bara godkända användare får tillträde till ett område eller en tjänst.

Något som länge har varit aktuellt är Flash till iPhone OS. I oktober 2009 lanserade Adobe version 10.1 av sin Flash Player och den anpassades för en rad olika smartphones.[56]
Utgångspunkten är att alla ska kunna använda Flash var de än är och de operativsystem som inkluderas är Symbian, Android, Palms Web OS och Windows Mobile.

Eftersom Apple inte har löst problemet med Flash Player så har Adobe löst problemet själva genom att i webbläsaren Safari i iPhone OS som inte stödjer innehåll som kräver Flash, kommer däremot utvecklare kunna skapa vanliga iPhone-applikationer med hjälp av Flash. Då exporterar man helt enkelt sin skapelse, kanske främst spel, så att de blir en applikation som sedan är möjlig att erbjuda kunderna via Apples Appstore.

[55] http://mashable.com/2009/07/10/iphone-object-recognition/
[56] http://labs.adobe.com/technologies/flashplayer10/

6 Godkännande av en applikation

Att få en applikation godkänd av Apple är en lång och utdragen process. För att få en möjlighet att bli godkänd måste man följa en rad procedurer.

Först ska man registrera sig som användare på iPhone Development Center, som är Apples portal för utveckling av iPhone. Där kan man ladda ner SDK, och den mjukvara som behövs för utveckling och testning. För att få skicka in en applikation måste man köpa en utvecklarlicens. Den kostar 99 dollar och ger dig licenser för att testa på ett begränsat antal telefoner. När allt är betalt och du har blivit godkänd av Apple som utvecklare är det bara att sätta igång att utveckla din egen applikation.

När din egen applikation är klar är det dags för lansering. För att få ut den på iTunes måste man godkänna att Apple tar 30 % av intäkterna.

Därefter skall man fylla i information om applikationen såsom namn, versionsnummer och beskrivningar. Tillslut kommer man till uppladdningen av applikationen till Apple.
Nu återstår det bara att vänta på att Apples utredning av applikationen skall bli godkänd. Från Apples sida kollar man på användarsäkerhet och pålitlighet i programmet. Man kollar sedan noggrant på att man inte konkurrerar med Apples redan befintliga applikationer.

Några saker som brukar ställa till problem, enligt oss själva, för att få applikationen godkänd är följande:

Dålig planering: Om man skriver en nätverksapplikation som samverkar med en server, och man passerar för mycket jobb till servern istället för att hantera det i applikationen. Även kan man få problem om man har skapat en applikation som begär för mycket data från servern och på så vis blir applikationen för datakrävande.

Dålig säkerhet: Alla sorters säkerhetsbrister stoppas direkt av Apple. Din applikation ska klara av de massiva tester som Apple utför.

Om applikationen blir godkänd läggs den ut på iTunes för försäljning.

Del 3

7 Diskussion

Att på en mobil, med mycket begränsat minnesutrymme, inte säkerhetsställa minneshantering är dumdristigt. Vi som programmerare har ett stort ansvar att vara kostnadseffektiva och inte orsaka systemkrascher. Därför vill vi genom den här rapporten utöka förståelsen bland programmerare för svårigheter som dyker upp vid programmering av applikationer för iPhone med hjälp av metoder och exempel för minneshantering. Vi har valt att gå djupare in på hur minnet påverkar framtagning och utveckling av applikationer. Hur skall programmeraren göra för att inte riskera en säkerhetsbugg? Vad är viktigt att tänka på för att undvika programkrascher?

Allokering av minne

Precis som man kan läsa i kapitel *5.2.2* måste man vara insatt i hur hantering av objekt fungerar. Om man inte tar hand om sina allokerade minnesplatser, kommer så småningom minnet att ta slut. Hur fungerar det här i praktiken?
Något som gör minneshanteringen extra viktigt för applikationer till iPhone är inte bara att en telefon inte är utrustad med lika mycket internminne som en stationär dator utan att det heller inte finns någon skräpsamlare, en så kallad *Garbage Collector*. Detta gör att man alltid måste ta hand om sin egen minnesallokering.

Det krävs två steg för att skapa ett objekt i Objective-C

- Dynamiskt allokera minne för objektet
- Initialisera den nyligen allokerade minnet till lämpligt värde

Ett objekt är inte fullt fungerande tills båda stegen har slutförts. Här nedan visas två exempel på hur man implementerar det på en eller två rader kod:

Ex1.
```
id anotherObj = [NSObject alloc]; //Allokera minne
anotherObj = [anotherObj init];  //Initiera minnet.
```

Ex 2.
```
id oneMoreObj = [[NSObject alloc] init]; //Allokera och initiera på samma rad.
```

Sammanfattningsvis måste programmeraren först se till att allokera minne och sedan initiera det. Det separerade anslaget från initiering ger programmeraren individuell kontroll över varje steg så att det ena steget kan modifieras oberoende av den andra.

I minnet i Objective-C fördelas nya objekt med hjälp av klassens metoder som fastställs i *”NSObject-klassen”*. *”NSObject”* definierar två huvudsakliga metoder för detta ändamål, *”Alloc”* och *”allocWithZone”*.

Dessa metoder allokerar tillräckligt mycket minne för att hålla alla instansvariabler för objektet, vilket tillhör den mottagbara klassen. Objekten behöver inte åsidosättas eller ändras i

underklasserna.

”Alloc” och *”allocWithZone”* metoder initierar det nyligen tilldelade objektets instansvariabel så att den pekar på objektets klass. Alla andra instansvariabler är satta till värdet noll. Vanligtvis måste ett objekt vara mer specifikt initierat innan det kan användas säkert. Denna initiering är ansvarig för mer klasspecifika exempelmetoder som börjar med förkortningen "*init*". Om metoden inte tar några argument, får metodnamnet vara *”init”*. Om metoden tar argument blir namnet "*init*" följt av prefix. Exempelvis kan ett *”NSView”* objekt initieras med en *”initWithFrame”* metod. Varje minnesallokering har en prestandakostnad. Denna kostnad ingår i den tid det tar att fördela minnet i programmets logiska adressutrymme och den tid det tar att tilldela programmet dess fysiska adress. Man rekommenderar att inte allokera minnet vid laddning utan vänta tills man absolut behöver det, allt för att minska onödiga beräknings- och väntetider. Istället bör man fokusera minnesallokeringen på det som krävs för att visa användargränssnittet och svara på input från användaren. Vänta med andra tilldelningar tills användaren skickar input om att det behövs mer minnesallokering.

Felhantering vid minnesproblem

Varningar
I Objective-C finns det en metod som heter *”didReceiveMemoryWarning”*. När den metoden anropas, undersöker den om ”*view controller*” kan släppa sina ”*views*”. Detta kommer att ske om ”*viewen*” inte har någon ”*superview*”. Varningen ”*didReceiveMemoryWarning*” ligger och söker efter signaler från operativsystemet ifall minnet skulle börja ta slut. Ifall man får en varning om att minnet är på väg att ta slut kommer metoden att köras. Detta sker automatiskt och minskar chanserna för att ett program kraschar. Eftersom man försöker frigöra all minnesallokering förutom den som den aktuella ”*viewen*” använder sig av blir det oftast tillräckligt med minne över så att man kan fortsätta och köra programmet. De problem som kan uppstå vid den här sortens hantering av minne är att dels tar själva processen upp minne och sedan om *”didReceiveMemoryWarning”* utför sina åtgärder och tar bort ”*views*” som inte används, är det inte säkert att det räcker. Om ditt program fortsätter läcka minne kommer det snabbt ta slut och programmet kommer slutligen att krascha.

Fallgropar
Vanliga fallgropar när man jobbar med allokering är att man vid multipla trådar anropar funktioner som allokerar minne vid varje anrop. Vi testade att köra följande kod:

```
MyTest* GetGlobalBuffer()
{
    static MytTest*  sGlobalBuffer = NULL;
    if(sGlobalBuffer == NULL)
        {
            sGlobalBuffer = malloc( sizeof(MyTest));
        }
        returjn sGlobalBuffer;
}
```

Om man anropar den här funktionen vid multipla trådar kommer det för varje anrop att allokeras minne lika stort som ”*sizeof(MyTest)*”. Detta gjorde att programmet kraschade och

man istället var tvungen att skapa metoden global. På så vis allokerar man minnet endast en gång och tar inte upp onödigt mycket plats.

8 Slutsats

Att skriva en rapport om en produkt från Apple är spännande och en otrolig utmaning. Men att få ut information från ett så pass slutet företag som Apple är inte lätt. Vi har lagt ner stor tid på att hitta alternativa källor till Apples egen dokumentation, dock utan större framgång. Det finns ytterst få källor som inte hänvisar, eller är direkt tagna från developer.apple.com, som är Apples API.

Mål med den här rapporten var att man som van programmerare för plattformar skulle kunna läsa den och på så vis kunna dra nytta på ett sätt som gynnar deras programmering för iPhone. Vi har försökt att göra en allmän inledning som sedan har smalnat av och riktat in sig på minneshantering på en iPhone.

I början av vår rapport tar vi upp de konkurrenter som finns samt hur marknaden ser ut just nu och vilka aktörer som finns. Det hjälpte oss att dra slutsatsen att många företag försöker hävda sig mot Apple och kopiera iPhone. Men man kan även utläsa att det finns en betydligt större öppenhet, hos exempelvis Google och deras Android telefoner. Där är man villig att släppa dokumentation från internutveckling och man väntar inte med att publicera allt tills produkten är släppt. Men så här är det när det gäller iPhone och Apple och det är bara att acceptera.

När den här rapporten stiftades så var målet att man skall läsa den och få tips och kunskap om saker som gör att programmeringen till iPhone applikationen skulle gå lättare och bli bättre. Man kan, om man har grundläggande kunskaper inom programmering, lära sig saker som är svåra att hitta information om. Vi började lägga upp problemet alldeles för stort och allmänt och trodde att man kunde sammanfatta allt som är bra att tänka på i en enda rapport. Det gick inte och vi var tvungna att avgränsa oss mot ett mål. Då vi ansåg att minneshantering är en viktig och intressant del valde vi att koncentrera oss på det.

Minneshanteringen på iPhone är inte likadan som på Mac OS X. Man har ingen skräphantering, vilket innebär att man måste stå för all allokering själv. Man har heller inte den mängd minne som vid en dator.

Pekare i Objective-C är kraftfulla, men är samtidigt en tidstjuv. Man lägger ner massor med tid och kraft för att inte få ett dåligt upplägg av programmeringen. Men minneshanteringen tar massor med energi från själva programidén. Varför ska jag som programmerare behöva hålla ordning på all allokering och release?

En annan störning av Objective-C är de mönster som måste följas: tillämpning av korrekt ”*init- och dealloc-metoder*” är ett måste för att programmet skall fungera.
Man kan undra varför ”*Alloc*” och ”*init*” måste vara separerade. Varför måste man skriva *”[[aloc Foo] initWithArg: arg]”?* Varför inte bara *”[nya Foo ARG]”?* Eller *nya ”Foo (ARG)”*, som det skrivs i språket Java!

När man summerar tankar och funderingar över det material och de tester vi har gjort för att framställa den här rapporten kan man definitivt säga att minneshantering är viktig och oerhört svårt. Man behöver hela tiden, på ett sätt som inte förekommer i bland annat Java, tänka på allokering och release lika mycket som i Objective-C. Varför är det så? Vi har inget bra svar, mer än att kraftfullheten väger över, tycker Apple. Vi ställer oss dock mycket mer tveksamma till hanteringen av den här frågan. Kanske hade man underlättat för gemene programmerare att skippa hela den här biten, men nu är det inte så och då får man ta det och koncentrera sig på att göra rätt redan från början.

Att programmera till iPhone är oerhört kul! Vi pratar mycket om minneshantering och dess negativa sidor, men det är bara en del av själva programutvecklingen. Helheten är bra och applikationsmarknaden för iPhone är kanske den hetaste just idag.

Vi hoppas att med hjälp av den här rapporten skapa en större förståelse för er läsare som inte har programmerat till iPhone innan. Genom att få en bakgrund, både om programmering för Mac OS X och till iPhone och dess API, också få en större helhetsbild som kommer att gynna programkvalitén.

9 Referenser

9.1 Internet

- Adobe. (2010). *Adobe Flash Player 10.1*. 2008-11-19 [www]. Hämtat från http://labs.adobe.com/technologies/flashplayer10/ Besökt 2010-04-01

- Andersson, M. (2008). *Apple Iphone blir mer företagsvänlig.* 2008-03-07 [www]. Hämtat från http://www.datormagazin.se/nyheter/article213804.ece Besökt 2010-03-01

- Android. (2010). *Android Timeline. 2010* [www]. *Hämtat från* http://www.android.com/about/timeline.html Besökt 2010-03-29

- Android. (2010). *What is Android?*. 2010-04-12 [www]. Hämtat från http://developer.android.com/guide/basics/what-is-android.html Besökt 2010-04-27

- Apache. (2010). *Licenses. 2010 [www] Hämtat från* http://www.apache.org/licenses/ Besökt 2010-04-20

- Apple. (2010). *Appstore. 2010* [www]. Hämtat från http://www.apple.com/ipad/app-store/ Besökt 2010-03-30

- Apple. (1996). *Apple Computer, Inc. Agrees to Acquire NeXT Software Inc.* 1996-12-20 [www]. Hämtat från http://web.archive.org/web/19970301172356/http://live.apple.com/next/961220.pr.rel.next.html Besökt 2010-03-12

- Apple. (2010). *Developer Tools Xcode.* 2010 [www]. Hämtat från http://developer.apple.com/technologies/tools/xcode.html Besökt 2010-03-15

- Apple. (2009). *Defining a Class.* 2009-10-19 [www]. Hämtat från http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Articles/ocDefiningClasses.html#//apple_ref/doc/uid/TP30001163-CH12-SW1 Besökt 2010-04-20

- Apple. (2009). *Fundamental Optimization Tips.* 2009-07-07 [www]. Hämtat från http://developer.apple.com/iphone/library/documentation/Performance/Conceptual/PerformanceOverview/BasicTips/BasicTips.html#//apple_ref/doc/uid/TP40001410-CH204-BBCIFICC Besökt 2010-03-20

- Apple. (2008). *Garbage Collection for Cocoa Essentials.* 2008-11-19 [www]. Hämtat från http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html Besökt 2010-04-15

- Apple. (2008). *Garbage Collection for Cocoa Essentials. 2008-11-19 [www]. Hämtat från* http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1 Besökt 2010-04-05

- Apple. (2008). *Garbage Collection for Cocoa Essentials.* 2008-11-19 [www]. Hämtat från http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/GarbageCollection/Articles/gcEssentials.html#//apple_ref/doc/uid/TP40002452-SW1 Besökt 2010-04-15

- Apple. (2010). Introduction. 2010-03-24 [www] Hämtat Från http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/CocoaFundamentals/Introduction/Introduction.html Besökt 2010-03-18

- Apple. (2009). *Introduction to The Objective-C Programming Language*. 2009-10-19 [www]. Hämtat från http://developer.apple.com/Mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html Besökt 2010-03-01

- Apple. (2010). *iPhone. 2010* [www]. Hämtat från http://www.apple.com/batteries/iphone.html Besökt 2010-04-20

- Apple. *Language Standards Supported by GCC.* [www]. Hämtat från http://developer.apple.com/mac/library/documentation/DeveloperTools/gcc-4.0.1/gcc/Standards.html Besökt 2010-03-13

- Apple. (2010). *Mac OS X*. 2010 [www]. Hämtat från http://www.apple.com/macosx/ Besökt 2010-02-15

- Apple. (2007). *Apple Reinvents the Phone with iPhone.* 2007-01-09 [www]. Hämtat från http://www.apple.com/pr/library/2007/01/09iphone.html Besökt 2010-03-10

- Apple. (2010). *Setting Required Hardware Capabilities.* 2010-03-24 [www]. Hämtat från http://developer.apple.com/iphone/library/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AdvancedFeatures/AdvancedFeatures.html Besökt 2010-04-15

- Apple. (2010). *Using iPhone Simulator.* 2010-03-19 [www]. Hämtat från http://developer.apple.com/iphone/library/documentation/Xcode/Conceptual/iphone_development/125-Using_iPhone_Simulator/iphone_simulator_application.html Besökt 2010-04-15

- Apple. (2008). *What Is Cocoa?.* 2010-03-24 [www]. Hämtat från http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html Besökt 2010-04-15

- Apple. (2010). *Why you'll love iPhone.* 2010 [www]. Hämtat från http://www.apple.com/iphone/why-iphone/ Besökt 2010-01-27

- Arstechnica. (2009). *Windows Mobile loses 2.7% of smartphone market in Q209.* 2009-08-12 [www] Hämtat från http://arstechnica.com/microsoft/news/2009/08/windows-mobile-loses-27-of-smartphone-market-in-q2.ars Besökt 2010-03-15

- Augustsson, T. (2010). *Iphone ett lyft för Apple – igen.* 2010-01-27 [www]. Hämtat från http://www.e24.se/lifestyle/prylar/iphone-ett-lyft-for-apple-igen_1820489.e24 Besökt 2010-01-28

- Blackberry. (2010). *Advanced Features. 2010* [www]. Hämtat från http://na.blackberry.com/eng/developers/browserdev/advfeatures.jsp Besökt 2010-04-10

- Bunch, K. (2009). *Apple IPhone Revolution.* 2009-01-02 [www]. Hämtat från http://www.kennybunch.com/blog/2009/01/apple-iphone-revolution/ Besökt 2010-01-27

- comScore. (2010). *Use of Social Media via Mobile Sees Considerable Gains in Past Three Months.* 2010-03-10 [www]. Hämtat från http://www.comscore.com/Press_Events/Press_Releases/2010/3/comScore_Reports_January_2010_U.S._Mobile_Subscriber_Market_Share Besökt 2010-03-12

- Dennisthe2. (2010). *History of the iPhone.* 2010-03-28 [www]. Hämtat från http://en.wikipedia.org/w/index.php?title=History_of_the_iPhone&oldid=359785831 Besökt 2010-04-15

- GSM Arena. (2010). Apple iPhone 3GS. (2010) [www] Hämtat från http://www.gsmarena.com/apple_iphone_3gs-2826.php Besökt 2010-03-17

- Ingdahl, W. (2009). *Hur starkt kommer mobiltelefonimarknaden att utvecklas?.* 2009-08-21 [www]. Hämtat från http://skiften.se/2009/08/21/hur-starkt-kommer-mobiltelefonimarknaden-att-utvecklas/ Besökt 2010-01-27

- Kerris, N. (2010) *Apple Reinvents the Phone with iPhone.* 2010 [www]. Hämtat från http://www.apple.com/pr/library/2007/01/09iphone.html Besökt 2010-02-03

- Macpro. (2009). *Macprointervjun: Aggressive Development.* 2009-03-12 [www]. Hämtat från http://www.macpro.se/2009/03/macprointervjun-aggressive-development/ Besökt 2010-01-27

- Microsoft. (2010). *Differences Between the .NET Compact Framework and the .NET Framework.* 2010 [www]. Hämtat från http://msdn.microsoft.com/en-us/library/2weec7k5.aspx Besökt 2010-04-12

- Motorola. (2005). *Motorola ROKR.* 2005 [www]. Hämtat från http://www.motorola.com/motoinfo/product/details.jsp?globalObjectId=117 Besökt 2010-01-27

- OHA. (2010). *Overview. [www} Hämtat från* http://www.openhandsetalliance.com/oha_overview.html Besökt 2010-03-11

- Ola. (2010). *Är iPhone det nya Windows?.* 2009-11-03 [www]. Hämtat från http://ohsohightech.se/ar-iphone-det-nya-windows/ Besökt 2010-01-27

- OpenGL. (2010). *OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. 2010* [www]. Hämtat från http://www.khronos.org/opengles/1_X/ Besökt 2010-04-20

- OpenGL. (2009). *OpenGL SDK*. 2009 [www]. Hämtat från http://www.opengl.org/sdk/ Besökt 2010-03-25

- Personal Computer World. (2001). *The men who really invented the GUI.* 2001-10-04 [www] Hämtat från http://www.pcw.co.uk/personal-computer-world/features/2045763/men-really-invented-gui Besökt 2010-03-19

- Schroeder, S. (2009). *The Future of the iPhone: Intelligent Object Recognition.* 2009 [www]. Hämtat från http://mashable.com/2009/07/10/iphone-object-recognition/ Besökt 2010-04-28

- Simmons, Curt. (2004). *How to Do Everything with Your Blackberry*. 2004 [www]. Hämtat från http://site.ebrary.com.focus.lib.kth.se/lib/kth/docDetail.action?docID=10130499 Besökt 2010-04-06

- Singh, A. (2004). *The NeXT Chapter.* 2004-02 [www]. Hämtat från http://www.kernelthread.com/publications/appleoshistory/7.html Besökt 2010-02-26

- Stadigs, J. (2010). *Android snabbast växande mobiloperativet.* 2010-01-26 [www]. Hämtat från http://www.idg.se/2.1085/1.288649/android-snabbast-vaxande-mobiloperativet Besökt 2010-02-03

- Treetop. (2010). *Mobil utveckling för iPhone.* 2010-02-25 [www]. Hämtat från http://www.treetop.se/sv/Nyheter/Treetop-CV Besökt 2010-02-14

- Welch, J. (2007). *iPhone vs. Windows Mobile: And the Winner Is…*2007-02-01 [www]. Hämtat från http://itmanagement.earthweb.com/columns/appleent/article.php/3657946/iPhone-vs-Windows-Mobile-And-the-Winner-Is.htm Besökt 2010-01-27

- Zachrison, O. (2010). *Iphone - Svenssons nya mobiltelefon.* 2010-02-15 [www]. Hämtat från http://www.e24.se/business/it-och-telekom/iphone-svenssons-nya mobiltelefon_1865013.e24 Besökt 2010-02-14

9.2 Böcker

- Höst, M. Regnell, B. Runeson, P. (2006). *Att Genomföra examensarbete*. 1a uppl. Sverige: Studentlitteratur AB. 153 sidor. 978-91-44-00521-8.
  En liten bok som tar upp hur man genomför en examensrapport, från början till slut.

- Wagner, R. (2008). *Professional iPhone and iPod touch Programming.* USA: Wiley Publishing, Inc. 284 sidor. 978-0-470-25155-3.
  En Bok som handlar om hur man programmerar i iPhone, iPod touch och mobile safari