HANDELSHÖGSKOLAN
VID GÖTEBORGS UNIVERSITET
INSTITUTIONEN FÖR INFORMATIK

MAGISTERUPPSATS I INFORMATIK 20 POÄNG

JAVA 2 MICRO EDITION - FÖR SVERIGE I MOBILEN?

Andreas Carlsson & Lars Setterlund

HT 2002

Inom en snar framtid kommer de flesta mobila terminaler ha stöd för utbytbara program. Med mobiltelefoner kommer det gå att göra mycket mer än att utföra telefonsamtal och sända textmeddelanden. Målet med denna uppsats är att utreda om Java™ 2 Micro Edition (J2ME) kommer att bli standard för nerladdningsbara applikationer för telefoner. Med Javateknik i mobiltelefonerna kan användare i större utsträckning än tidigare bestämma innehållet i den egna telefonen och på det viset göra telefonen personligare. Nya applikationer kan laddas ner från operatörernas portaler eller från andra platser på internet. Spel förväntas i ett initialt skede dominera marknaden men med tiden kommer andra applikationer och tjänster att ta marknadsandelar. Vår kvalitativa studie bygger på intervjuer, litteraturstudier och praktiskt programmeringsarbete. Uppsatsen utreder vad branschen har för intentioner med Java för mobil-telefoner. Den primära slutsatsen av uppsatsen är att Java 2 Micro Edition och profilen MIDP kommer att vara en de facto standard på mobiltelefoner inom några år.

Handledare:
Lennart Peterson
Universitetsadjunkt/Systemadministratör vid
Institutionen för Informatik, Göteborgs Universitet

Förord

Vi vill tacka de personer som på olika sätt gjort det möjligt för oss att skriva denna uppsats:

Intervjupersoner:
Tack för att ni varit så tillmötesgående. Utan er, ingen uppsats.

Lennart Peterson:
Tack för en avslappnad handledning och snabb hjälp med utvecklingsmiljöer.

Göteborg, januari 2003

Andreas Carlsson Lars Setterlund

Figurlista

1 Inledning

Mobiltelefonen är en av de mest uppskattade uppfinningarna i dagens samhälle vilket i mångt och mycket beror på människans behov av kommunikation. Med hjälp av mobiltelefonen kan människan ta emot och sända information i stort sett var hon än befinner sig. I början var mobiltelefonen bara en telefon men numera har den fått rollen som en kombinerad kommunikation- och mediacentral. Förutom att tala i telefon går det att skicka multimedia och textmeddelanden, spela spel, skicka e-post, lyssna på musik och surfa på internet. Numera går det dessutom att hämta ner nya program och nyttja tjänster på ett sätt som tidigare inte var möjligt.

I och med att mobil kommunikation breder ut sig allt mer i samhället, ökar marknaden för allehanda tjänster. Att erbjuda programvara till de olika handhållna enheter som finns på marknaden verkar inte vara en lätt uppgift. Svårigheten är att hårdvaran varierar vad gäller minne, processorer och skärmar. Beträffande operativsystem finns varken standard eller de facto standard att luta sig mot. Så i vilket programmeringsspråk skall man utveckla program? Och för vilket operativsystem?

Traditionellt sett har en mobiltelefon ett antal förinstallerade applikationer vid inköp. Tillverkarna väljer omsorgsfullt ut vilka program konsumenterna skall få tillgång till, inte minst för att locka till sig kunder. Telefontillverkarna kommer förmodligen alltid att erbjuda ett antal grundfunktioner. Med Javateknik anpassad för mobiltelefoner kan program laddas ner när som helst till telefonen. Användaren kan anpassa sin telefon efter tycke och smak. Det kan gälla nya spel, en kalender eller shoppinglista där flera personer har tillgång till samma information. Andra exempel är en engelsk-svensk ordlista, program för att sköta banktjänster, eller vad sägs om att kunna ha en zoomningsbar karta i telefonen som visar användarens position. Möjligheterna kanske inte är oändliga, men de är minst sagt omfattande.

För att nya applikationer i stor utsträckning skall bli tillgängliga för allmänheten måste det vara lätt för utvecklarna att skriva program för plattformen. Java är ett programmeringsspråk som behärskas av många programmerare och språket är förhållandevis lätt att lära sig. Dessa faktorer är viktiga när det gäller att tillgodose marknaden med ett stort antal applikationer. Frågan är dock om Java har de förutsättningar som krävs för att bli en standard när det gäller extraapplikationer för mobiltelefoner. Uppsatsens mål är att besvara

den frågan. Förhoppningen är att läsaren genom uppsatsen skall kunna bilda sig en bra uppfattning om Java 2 Micro Edition (J2ME) och Mobile Information Device Profile (MIDP), vilka är de delar av Java som oftast används när det gäller telefoner. MIDP beskrivs utförligare i teoriavsnittet under rubriken Mobile Information Profile. Utöver det tekniska, skall uppsatsen ge en fingervisning om åt vilket håll tekniken och branschen är på väg.

1.1 Problemområde

Mobiltelefoner är ett av många områden där utvecklingen går snabbt framåt. Det beror mycket på att det inte är en mogen teknik och att tillverkarna måste överträffa varandra med tekniska små underverk gång efter annan för att locka till sig konsumenterna. Nya tekniker och tjänster har skapats såsom: Short Message Service (SMS), Wireless Application Protocol (WAP), General Package Radio Service (GPRS), Enhanced Message Service (EMS) och Multimedia Message Service (MMS). Vissa av tjänsterna har tagits emot av användarna med öppna armar medan andra har fått ett kallsinnigare mottagande. Trenden för dagens telefoner är att de får lite större skärmar med färg, mer minne, snabbare processorer, extra exekveringsmiljö och snabbare uppkoppling för att ta emot och sända data. Dessa egenskaper öppnar upp för nya sorters applikationer för mobiltelefoner. Program kan hämtas till telefonen över telefonnätet (Over The Air (OTA)) för att senare installeras och köras på telefonen. Problemet är att det finns flera alternativa tekniker för att göra ungefär samma sak. Branschen som består bland annat av nätverksoperatörer, telefontillverkare, applikations- och plattformsleverantörer drar kanske inte alltid åt samma håll. Vilken eller vilka tekniker som kommer att bli framgångsrika beror på vilka val som görs av de stora aktörerna och hur tjänsterna tas emot på marknaden.

1.2 Problemformulering

Uppsatsen skall försöka besvara nedanstående frågeställningar, analysera svaren och förutspå framtiden för J2ME på mobiltelefonmarknaden.

- Hur utvecklas program i J2ME för mobiltelefoner?
- Hur sker testningen?
- Vilka alternativa tekniker finns till J2ME?
- Vilka möjligheter och begränsningar finns för J2ME/MIDP?
- Vilken hänsyn måste tas till det begränsade minnesutrymmet när programmet skrivs?

- Vilka är skillnaderna i resultat och prestanda mellan tester i en simulerad miljö, och i de telefoner som skall använda sig av programmet?
- Vilka kommer att leverera innehåll?
- Åt vilket håll är J2ME på väg?
- Vilka nya möjligheter kommer med MIDP2.0?
- Hur ser samarbetet ut mellan operatörer, telefontillverkare och applikationsleverantörer?
- Vilka driver utvecklingen framåt?
- Vem skapar tekniken och i vilket syfte?

1.3 Syfte

Under sommaren och hösten 2002 lanserades ett flertal telefonmodeller med stöd för Java, vilket väckte ett intresse för J2ME hos författarna. Syftet med uppsatsen är att utreda huruvida Java kommer att vara den plattform som kommer att dominera marknaden för nerladdningsbara applikationer.

1.4 Avgränsning

Då J2ME sträcker sig över en ganska stor skara enheter, allt från personsökare till informationsterminaler, har uppsatsen inriktats mot mobiltelefoner och MIDP (den profil som används på telefoner). Övriga profiler inom J2ME berörs mer översiktligt för att läsaren skall få en helhetsbild av J2ME. Det förutsätts att läsaren av denna uppsats redan är insatt i Javatekniken (information finns på http://java.sun.com). Uppsatsen har inriktats på hemmamarknaden, det vill säga norra Europa med fokus lagd på Sverige.

2 Teori

Det teoriavsnitt som följer, är till för att ge läsaren av uppsatsen en god inblick i Java för mobiltelefoner.

2.1 Java

Java är ett objektorienterat programspråk som utvecklas av Sun Microsystems Inc (Sun). Java skapades med intentionen att få fram ett enkelt, säkert och portabelt programmeringsspråk. Språket påminner syntaxmässigt om C och C++ men den stora skillnaden är att Javakod inte kompileras till maskinkod utan till Javabytekod. Bytekoden tolkas av en virtuell maskin (VM) som översätter den till maskinspecifika

instruktioner. Detta innebär att programmen kan exekveras på vilken dator som helst under förutsättning att det finns en virtuell maskin för Java tillgänglig. En av Javas starka egenskaper är att en stor del av programmeringen redan är gjord i form av olika klasser som tillhandahåller önskad funktionalitet. Dessa klasser återfinns i bibliotek kallade Application Programming Interface (API). [49]

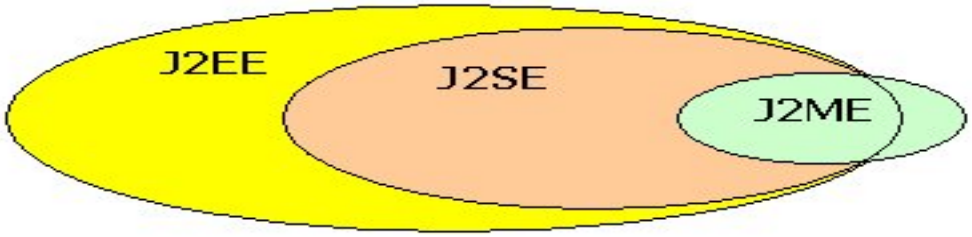
En av grundtankarna när Java utvecklades av Sun var ”WRITE ONCE, RUN ANYWHERE™”. I takt med att Java utvecklats och expanderat har man funnit behov av att dela upp språket i olika delar, vilket har lett till att grundtanken i viss mån har fått överges. Uppdelningen beror på att behoven på serversidan inte motsvarar behoven för handhållna enheter. Delarna kallas editions. [30][36]

Java™ 2 Platform, Standard Edition (J2SE)
J2SE är själva grunden i Java, uppbyggt för att köras på arbetsstationer och persondatorer. [30]

Java™ 2 Platform, Enterprise Edition (J2EE)
J2EE har inbyggt stöd för Servlets, Java Server Pages (JSP), och Extensible Markup Language (XML). J2EE är ämnat för serverbaserade applikationer. [30]

Java™ 2 Platform, Micro Edition (J2ME)
J2ME är designat för enheter med begränsad skärmstorlek, processorkraft och minne. [30]

Trots att språket är uppdelat i editions, delas vissa centrala klasser. Figur 1 illustrerar att J2ME både är en delmängd I J2SE och dessutom har fått tillägg som ligger utanför de båda andra. [30]



Figur 1 Hierarki för Javaplattformen

2.2 Java Community Process (JCP)

JCP är ansvarig för utvecklingen av Javatekniken. Det är en öppen organisation med en inre grupp av aktiva medlemmar som med hjälp av förslag från den intresserade allmänheten utvecklar och godkänner Javas tekniska specifikationer. Vem som helst kan bli medlem i JCP och ha en roll i processen. Arbetet med Java Community under JCP bidrar till att Javateknikernas standard, vad gäller säkerhet och plattformsoberoende, är fortsatt hög då intressenterna företräder olika organisationer och företag. Detta i sin tur leder till att Javaprogram kan exekveras på en skiftande flora av enheter, från persondatorer och industrirobotar till olika konsumentprodukter såsom mobiltelefoner och handdatorer. [20]

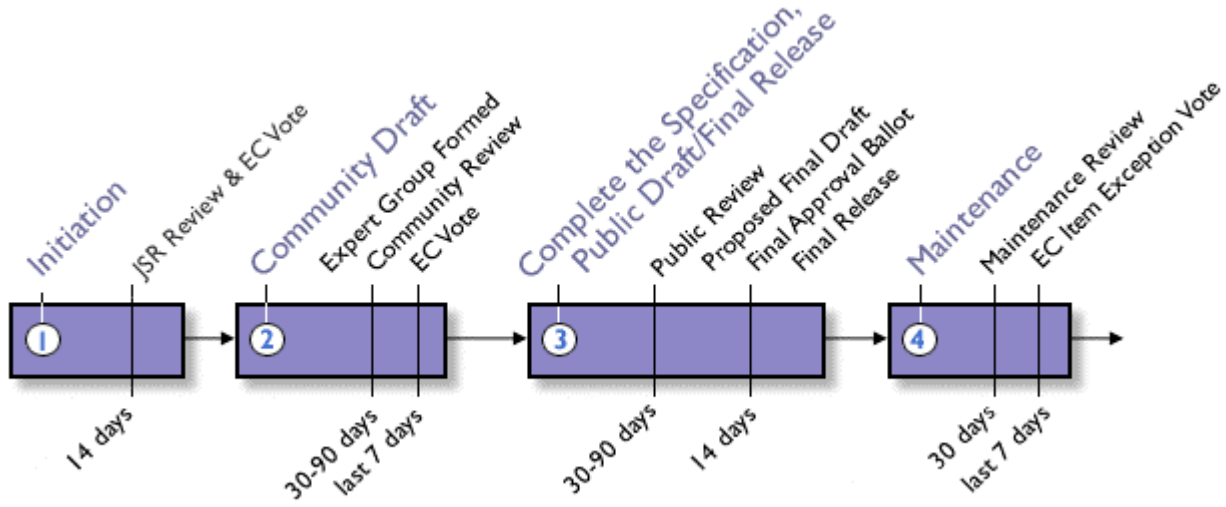
Arbetsgången inom JCP:
Steg1 - Initiering
Processen startar med att ett förslag till en ny specifikation, det vill säga en Java Specification Request (JSR), skickas till Process Management Office (PMO). JSR:er får endast skickas in av medlemmar i Java Community. Ett styrande organ kallat Executive Committee (EC) avgör om JSR:en skall ha en plats inom Java och utvecklas under JCP. Det finns två EC, en för J2SE/J2EE och en för J2ME. [19][40]

Steg2 – Internt utkast
En grupp av experter utses av EC. Denna expertgrupp formar ett första utkast av specifikationen som sedan EC, expertgruppen och övriga medlemmar i JCP får lämna synpunkter på. Kritiken används av expertgruppen för att ändra och förfina utkastet. Till sist avgör den ansvariga EC:n om nästa steg skall utföras. [19]

Steg 3 - Publikt utkast
Utkastet lämnas ut till allmänheten för granskning. Vem som helst med uppkoppling mot internet kan läsa och lämna kommentarer på utkastet. Expertgruppen använder kommentarerna för att ytterligare förfina dokumenten. Till slut ser expertgruppens ledare till att en referensimplementation, dokumentation och testverktyg tas fram och skickas till EC:n för slutligt godkännande. [19]

Steg 4 – Underhåll
Den färdiga specifikationen uppdateras för att klargöra, utveckla och ändra i referensimplementation, dokumentation och testverktyg där det behövs. Detta görs av ledaren för expertgruppen. Den ansvariga EC:n granskar föreslagna ändringar till specifikationen och pekar på

vilka förslag som kan genomföras omedelbart och vilka som behöver genomgå granskning av expertgruppen. [19]



Figur 2 Java Community Process – Tidslinje [21]

2.3 Javas virtuella maskiner

En virtuell maskin emulerar en fysisk maskin genom att tolka instruktioner som skrivits i ett visst format. För Java är formatet bytekod. Ett Javaprogram tolkas av en virtuell maskin som översätter koden till maskinspecifika instruktioner, vilka utförs på värdenheten. På J2EE, J2SE och inom vissa delar av J2ME används virtuella maskiner som följer Java Virtual Machine Specification (JVMS). Till mobila enheter som på grund av begränsade resurser inte kan använda en virtuell maskin som stödjer allt som finns specificerat i JVMS, har det tagits fram en kompakt variant kallad Kilo Virtual Machine (KVM). KVM är designad så att den får plats i det begränsade minnesutrymmet som till exempel en mobiltelefon har. [30][54]

2.4 J2ME

J2ME står på tre hörnstenar: konfigurationer, profiler och extra paket. Konfigurationen definierar en Javaplattform för en grupp produkter med liknande förutsättningar gällande minne, display, nätverkskoppling och processorkraft. [2]

2.5 Konfigurationer

Konfigurationen ställer krav på den underliggande virtuella maskinen och på vilka API:er som den ovanliggande profilen implementerar. På figur 3 illustreras att det finns två konfigurationer för J2ME, Connected Device Configuration (CDC) och Connected Limited Device Configuration (CLDC). Skillnaderna mellan dessa är främst hur mycket minne de tar i anspråk för att köra Javas virtuella maskin, om nätverksuppkopplingen sker med hög respektive låg bandbredd och hur enheten strömförsörjs. En god tumregel är att enheter som är trådlösa använder sig av CLDC och de med sladdanslutning använder sig av CDC.

CDC är till för enheter med:

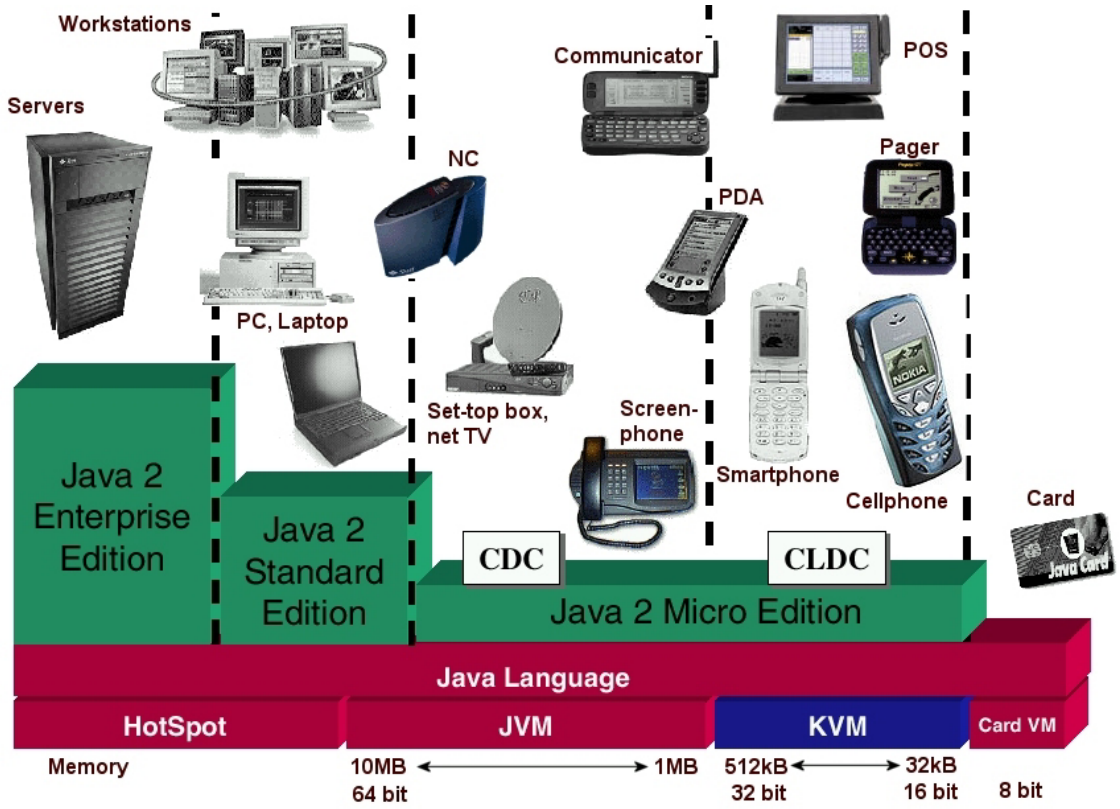
- Minst 512kb lagringsutrymme
- Minst 256kb arbetsminne
- Nätverksuppkoppling

CDC stödjer en fullständig implementation av Java Virtual Machine (JVM). [12]

CLDC är till för enheter med:

- Minst 128kb lagringsutrymme
- Minst 32kb arbetsminne
- Begränsade användargränssnitt
- Batteridrift
- Nätverksuppkoppling med begränsad bandbredd

På grund av de begränsade resurserna som CLDC specificerar, kräver den en virtuell maskin som är mindre resurskrävande än JVM. Denna gick till början under namnet Kuaui Virtual Machine men numera har Kuaui ersatts med Kilo. Namnet Kilo anspelar på att KVM kräver kilobytes snarare än megabytes för att implementeras på värdenheten. [11][45][17]



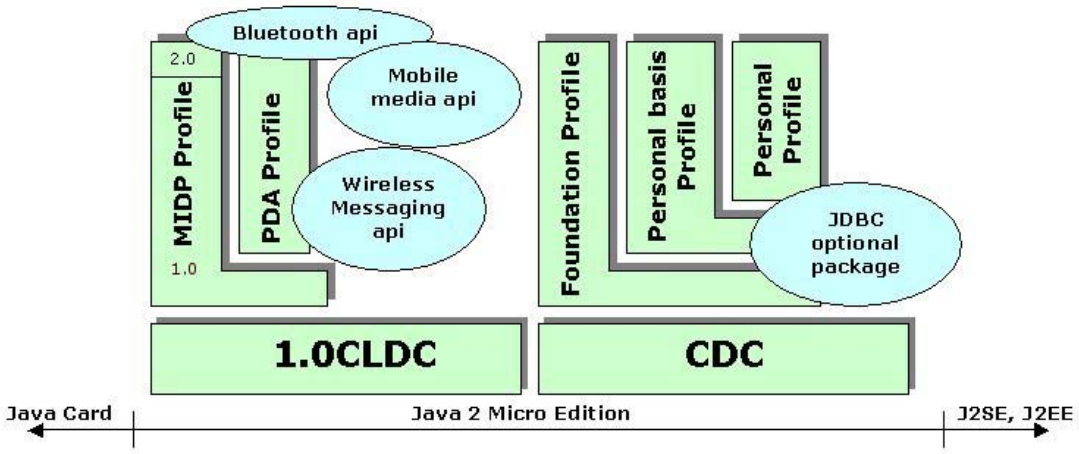
Figur 3 Översikt – Javas arkitektur

2.6 Profiler

Då skillnaderna är så stora mellan enheter som delar en konfiguration exempelvis gällande skärmstorlek och tillgängligt minne, har Sun valt att införa profiler för J2ME-plattformen. En profil kan sägas vara en utvidgning av en konfiguration som ger möjlighet för utvecklare att skriva applikationer för en speciell typ av enheter. Exempelvis definierar MIDP API:er för gränssnittskomponenter, inmatning av data, bestående minne och nätverksuppkoppling. Dessa definitioner tar i beaktande mobila enheters skärmstorlek och minnesbegränsning. Personal Digital Assistant Profile (PDA Profile) är anpassad för handdatorer och dess förutsättningar. Både MIDP och PDA Profile använder sig av CLDC. Övriga profiler för J2ME är bland andra Foundation Profile, Personal Profile och Remote Method Invokation Profile, vilka använder sig av CDC. Profiler definieras av arbetsgrupper bestående av företag som samarbetar i JCP. Samarbetet gör att företagens intressen tas tillvara när nya profiler utvecklas. [17][20][30]

2.7 Extra paket (Optional Packages)

Ett extra paket för J2ME har liksom profilerna ett API. Den stora skillnaden är dock att de extra paketen inte specificerar en komplett programmiljö. Ett extra paket används alltid ihop med en konfiguration och en profil. I paketen finns tillägg till profilerna som inte är tillräckligt generella för att passa inom en viss profil eller behöver finnas tillgängliga för flera profiler. Ett exempel är Wireless Messaging API (WMA) som innehåller klasser för att sända och ta emot SMS. Eftersom WMA är ett extra paket kan det användas ihop med alla J2ME-enheter som har stöd för SMS. [22][48]



Figur 4 Profiler och extra paket

2.8 Säkerhet

Alla enheter som kör Javaapplikationer behöver skydda sig mot ”elak” kod som kan komma åt systeminformation eller annan information. Detta gäller särskilt kod som laddats ner ”Over The Air” från öppna nätverk.

J2SE:s säkerhetsmodell består av tre lager:

- Javaspråket
- Javakompilatorn tillsammans med den virtuella maskinen
- Security Manager.

Säkerhetsegenskaper för Javaspråket:

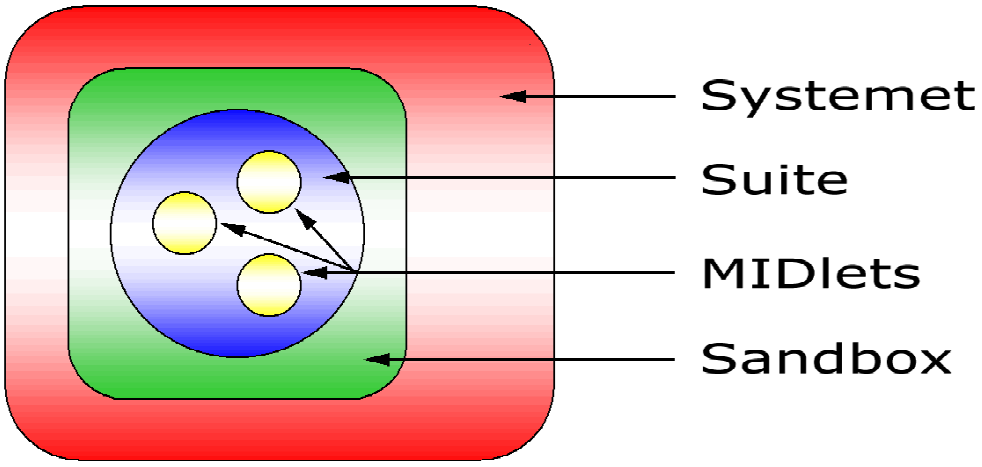
- Datatyper har samma storlek på alla plattformar. De är med andra ord inte maskinberoende.
- Pekararitmetik kan inte göras.

- Kontroll av storleken på fält (array) kastar ett ”exception” om man försöker indexera en post utanför fältets gränser.

Javakompilatorn och JVM ser till att systemet inte förstörs av illasinnad kod. Den virtuella maskinen skapar en säker omgivning som består av en klassverifierare (class-verifier), nerladdning av klasser och kontroll av bibliotek under exekveringstid samt en automatisk ”garbage collector”. Det är klassverifieraren som ser till att klassfiler inte exekveras på ett sätt som inte är tillåtet av JVM-specifikationen. Kortfattat kan man säga att Security Managern hindrar icke pålitlig kod att utföra de flesta operationer men tillåter pålitlig kod att utföra de flesta. [52]

J2SE:s säkerhetsmodell är dock inte applicerbar på CLDC/MIDP enheter, eftersom den kräver mycket minne. För att lösa detta problem delades klassfilsverifikationen upp i två steg, förhandsgranskning (pre-verification) och granskning i enheten (in-device verification). [52]

Förhandsgranskning sker med hjälp av utvecklingsverktyg och innebär att extra attribut tillsätts klassfilen. Dessa attribut är inget som i sig hindrar klasserna från att fungera under J2SE/J2EE. Klassfilerna blir ungefär 5 % större efter detta steg. Granskningen i enheten utförs av den virtuella maskinen som läser instruktionerna i klassfilerna, och kontroller görs för att validera koden. Granskningen kan när som helst rapportera ett fel och därmed också avvisa klassfilen. [52]



Figur 5 Sandbox för J2ME

Säkerhetsmodellen för CLDC/MIDP kallas för ”sandlådemodellen” och med det menas att koden exekveras inne i en avgränsad miljö. Applikationen har inte tillgång till objekt eller variabler utanför den avgränsade miljön, se figur 5. Detta uppnås genom att vissa delar av Javaspråket har eliminerats. Det gäller främst klasser som skulle kunna medföra säkerhetsrisker i frånvaron av en Security-Manager. De delar som är borttagna är:

- Java Native Interface (JNI). Man kan därmed inte komma åt andra programmeringsspråks API:er eller maskinspecifik kod, dels beroende på att det är osäkert men också på grund av att minnesanspråken för att tillhandahålla API:er är stora.

- Möjligheten att definiera egna ”klassladdare” (class loaders). En JVM som stödjer CLDC-specificationen har inbyggd ”klassladdare” som ej går ersätta, åsidosätta eller modifiera ”.

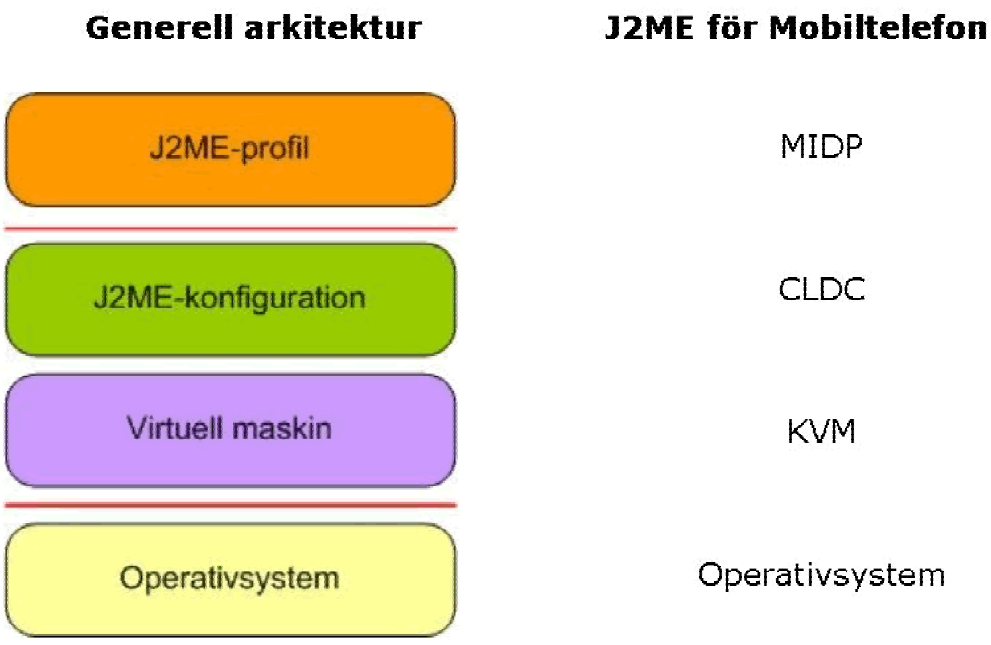
- Möjligheten att få information om vilka klasser som har laddats av den virtuella maskinen (reflections) och därmed stöds varken RMI eller serialisering (serialization).

- Gruppering av trådar.

- Svaga referenser (weak references). CLDC/MIDP applikationer kräver inte svaga referenser. [30][52]

2.9 Arkitekturen

I grunden finns enhetens operativsystem. Ovanpå operativsystemet ligger den virtuella maskinen som beroende på vilken enhet det är, antingen är JVM eller KVM. Den virtuella maskinen ger maskinspecifika instruktioner till enheten. I lagret över den virtuella maskinen ligger konfigurationen, i mobiltelefoners fall är det CLDC. Överst ligger enhetens profil. [30]



Figur 6 Arkitekturer

2.10 Kilo Virtual Machine

Sun:s virtuella maskin, som är en del av referensimplementationen för CLDC kallas KVM. För att uppfylla CLDC:s krav, är den utvecklad för att vara så liten som möjligt, men ändå kunna bevara alla centrala aspekter av Javaspråket. KVM kräver ett minimum av 128kb minnesutrymme inkluderat den virtuella maskinen, nödvändiga klassbibliotek, samt utrymme för att allokera objekt under en applikations körningstid. KVM är skriven i programmeringsspråket C för att den skall vara portabel och att så lite kod som möjligt skall behöva ändras för olika plattformar. Maskinberoende kod är begränsad till minneshantering, felhantering, kodinitialisering och uppstädning (garbage collection). [30][44]

2.11 Connected Limited Device Configuration – CLDC

En konfiguration är nära relaterad till den virtuella maskinen. Den virtuella maskinen ställer krav som måste tillgodoses av konfigurationen. CLDC definierar en standardplattform för små resursbegränsade nätverkskopplade enheter. När kraven sattes upp för CLDC var det viktigt att ta i beaktande olikheter i hårdvara på produkterna. Såväl processorer och minne som operativsystem skiljer

sig vida mellan de produkter som skall använda sig av konfigurationen. För att inte begränsa sig till ett fåtal produkter definierades få krav.
Krav på hårdvara:

- Minst 128kb minnesutrymme för den virtuella maskinen och för att lagra CLDC biblioteken. Detta minne måste bevara sitt innehåll även om enheten är avstängd.

- Minst 32kb arbetsminne för allokering av objekt under körning av program.

Krav på mjukvara:

- Att mjukvaran kan välja applikationer och starta applikationer.

- Möjlighet att kunna ta bort Javaapplikationer från enheten.

Det finns inga krav på att enheten skall klara av dynamisk nerladdning av applikationer. Om nerladdning inte stöds måste applikationerna installeras av tillverkaren, och i så fall är minneskraven lägre ställda.

Enheter som använder sig av CLDC har enligt dess JSR följande karaktär:

- Processorns klockfrekvens är minst mellan 8-32MHz.
- 16/32 bitars processor.
- Batteridrift.
- Nätverkskoppling med begränsad bandbredd.
- Massproducerade.
- Begränsade användargränssnitt.

På grund av de hårda krav som ställs av CLDC har många klasser uteslutits i CLDC-implementationen.

- Flyttalsberäkningar ställer höga krav på processorn och därför finns flyttal inte att tillgå när man programmerar efter CLDC-specifikationen.

- Finalize(). Denna metod stöds ej av CLDC på grund av att den är processorkrävande.

- KVM stödjer endast ett fåtal felhanteringsundantag. Skälen till detta är:
    - Omfattande felhantering, som i Standard Edition, är krävande för systemet.
    - Inbyggda system har ofta sin egen felhantering. Detta innebär att användaren ofta får trycka på ”reset-knappen”.

Den virtuella maskinen som stöds av CLDC skiljer sig från den som stöds av J2SE på följande vis:

- Flyttalsberäkningar saknar stöd då CLDC-specifikationen ej kräver flyttalsberäkning.

- Andra programmeringsspråks API:er och maskinspecifik kod är oåtkomlig. Ger ej stöd för Java Native Interface.

- CLDC kräver att den virtuella maskinen implementerar en ”laddare” (class loader). Denna class loader beskriver hur klasser laddas och hur fel hanteras. ”Laddaren” kan inte ersättas, åsidosättas eller modifieras. Denna ”loader” är definierad och implementerad av enhetens tillverkare.

- I J2SE kan information fås om vilka klasser som laddats av den virtuella maskinen, vilket inte går med den virtuella maskinen som CLDC stödjer.

- Den virtuella maskinen stödjer inte klassen ”ThreadGroup” och en grupp av ”trådar” kan inte startas endast genom ett metodanrop.

- Eftersom inte CLDC stödjer finalization gör inte CLDC:s virtuella maskin heller det.

[11][30]

2.12 Profiler

De olika profilerna har sina egna API:er som antingen är fristående (till exempel MIDP) eller bygger på någon eller några andra profiler (exempelvis Personal Profile). Det finns även API:er som är framtagna för att kunna användas ihop med flera profiler inom J2ME. Det är dessa som tidigare omnämnts som extra paket. Avsnittet beskriver övergripande de flesta av J2ME:s profiler. [30][48]

2.12.1 Mobile Information Device Profile

MIDP riktar sig mot handhållna enheter som har små skärmar, pekskärm eller knappsats och kan kommunicera över ett mobiltelefonnät med begränsad bandbredd. MIDP specifikationen kräver att skärmstorleken skall vara minst 96 pixlar (bildpunkter) bred och 54 pixlar hög. Utöver detta kräver specifikationen att skärmen skall kunna hantera minst två färger. Hur knappsatsen är utformad skiljer sig mycket mellan de olika enheterna. Specifikationen kräver ett minimum av knappar: siffrorna 0-9 tillsammans med navigationstangenter och en ”select-knapp”. MIDP läggs ovanpå CLDC i arkitekturen och tillhandahåller bland annat API:er för följande uppgifter:

- Skapa och styra applikationer. Applikationer skrivna för MIDP kallas MIDlets. De kan startas, pausas och förstöras i enlighet med den livscykel som beskrivs i MIDP-specifikationen.
- Visa text och grafik samt reagera på indata från användare.
- Spara data i en enkel databas (Record Management System (RMS)).
- Utföra uppgifter då en timer når ett visst värde.
- Nätverksansluta över HyperText Transfer Protocol (HTTP).

[13]

2.12.2 Foundation Profile

Foundation Profile har två uppgifter. Först och främst förser den J2ME med en profil passande för enheter vilka behöver en Javastödd nätverksmiljö som dock inte kräver ett grafiskt användargränssnitt. Dessutom är Foundation Profile en basprofil som kan användas av andra profiler som förutom det som specificeras i Foundation Profile även kan innehålla stöd för grafik eller annan funktionalitet. Denna profil bygger på CDC och skall alltså användas på enheter som når upp till CDC:s minimikrav. [14]

2.12.3 Personal Digital Assistant Profile (PDA Profile)

PDA Profile bygger på CLDC men utvidgar och omformar den lite. PDA Profile ger handdatorer en standardplattform för enheter som klarar dessa grundkrav:

- Minst 512KB (och maximalt 16 MB) totalt minnesutrymme (ROM + RAM) tillgängligt för Java körtidsmiljö och klassbibliotek.
- Batteridrift eller annan strömförsörjning
- Skärm med en upplösning på minst 20 000 pixlar, någon form av pekverktyg och möjlighet att skriva in text.

Till de klassbibliotek som ingår i CLDC lägger denna profil till API:er som innehåller en delmängd av Abstract Window Toolkit (AWT). Arbetet med PDA Profile är inte riktigt klart ännu, så huruvida användargränssnitten i MIDP kommer att ingå är ännu osäkert. [17]

2.12.4 RMI Profile

RMI Profile stödjer Java Remote Method Invokation (RMI). RMI används exempelvis för distribuerad beräkning, utskrifter på annan plats eller andra tjänster som inte kan eller bör utföras på den lokala enheten.

- Minst 2.5 MB ROM tillgängligt (minneskravet är ej tvingande).
- Minst 1 MB RAM tillgängligt (minneskravet är ej tvingande).
- Tcp/ip anslutning till nätet.
- Fullt stöd för Foundation Profile och CDC. [16]

2.12.5 Personal Profile

Personal Profile är till för enheter som behöver stabil uppkoppling mot internet. Den är tänkt att fungera som nästa generation av Sun:s Personal Java, och skall som sådan vara kompatibel med Personal Java-applikationer skrivna för J2SE 1.1.x och 1.2.x.
Dessutom skall enheterna som använder denna profil ha följande egenskaper:

- Minst 2.5 MB ROM tillgängligt (minneskravet är ej tvingande).
- Minst 1 MB RAM tillgängligt (minneskravet är ej tvingande).
- Stabil uppkoppling mot någon sorts nätverk.
- Grafiska användargränssitt vilka klarar av att köra Java applets.
- Har fullt stöd för J2ME Foundation Profile och CDC. [15][47]

2.12.6 Personal Basis Profile

CDC och Foundation Profile är den konfiguration och profil som Personal Basis Profile kommer att användas ihop med. Denna profil kommer att ersätta den plattform på vilken det i nuläget körs

minimala Personal Java applikationer. Personal Basis Profile kommer att vara lämplig för informationssystem i bilar och informationsstationer med begränsade användargränssnitt.

- Minst 2 MB ROM tillgängligt (minneskravet är ej tvingande)
- Minst 1 MB RAM tillgängligt (minneskravet är ej tvingande)
- Stabil uppkoppling mot någon sorts nätverk.
- Enkelt grafiskt användargränssnitt vilket klarar en delmängd av AWT.
- Fullt stöd för Foundation Profile och CDC. [23][47]

2.12.7 Java Game Profile

Java Game Profile inriktar sig mot spelutveckling. Även denna profil riktar in sig mot CDC och Foundation Profile. Det är speciellt nio områden som profilen inriktats mot:

- 3D-modellering och rendering för spel
- Fysisk 3D-modellering för spel
- 3D-animering av karaktärer i spel
- 2D rendering och buffertbyte av video för spel
- Spelordning och nätverkskommunikation
- Ljud för spel
- Spelkontroller
- Hårdvarutillgång för spel

Dessa områden ger grunderna för en spelplattform. Expertgruppen som står för utvecklingen av denna profil kommer i den mån det är möjligt att utnyttja befintliga API:er. För spelutvecklare är prestanda extremt viktigt, därför är det expertgruppens uppgift att komma på sätt att utnyttja hårdvaran på bästa sätt. Lyckas gruppen med detta är det mycket möjligt att denna profil även används inom J2SE, även om huvudsyftet med framtagandet är att skapa en bra profil för J2ME. [24]

2.12.8 Information Module Profile

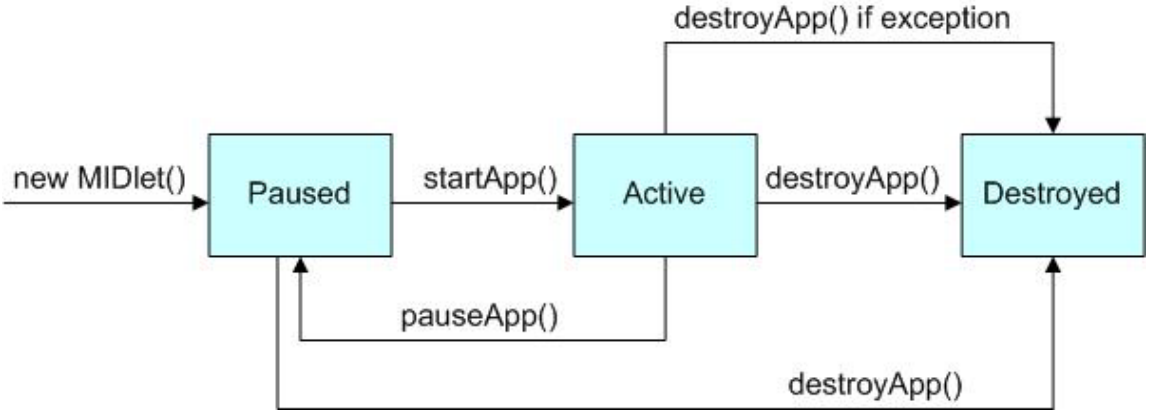
Information Module Profile kan ses som en delmängd av MIDP. Det som saknas är möjligheten att använda en display. Profilen är tänkt att användas ihop med CLDC och enheter såsom fjärrkontroller, modem, hushållsapparater, mätinstrument inom industrin och enheter i inbyggda system. Kraven för denna profil är desamma som för MIDP,

bortsett från allt som har med skärmar att göra. Profilen är under utformning. [10]

2.13 MIDlet

En MIDlet är en applikation byggd med den abstrakta MIDlet-klassen (javax.microedition.midlet.MIDlet) som superklass. En MIDlet saknar i likhet med Java Applets metoden main(). Kommunikationen mellan det underliggande operativsystemet och MIDleten styrs av en i enheten inbyggd mjukvara kallad Application Management Software (AMS). AMS kommunicerar med MIDleten genom de publika metoder som klassen måste implementera. Dessa metoder är främst startApp(), pauseApp() och destroyApp(). [30]

I en MIDlets livscykel ingår tillstånden pausad, aktiv, förstörd (paused, active och destroyed). En MIDlet befinner sig alltid i något av dessa tre tillstånd. En MIDlet placeras i paustillståndet efter att konstruktorn anropats. Därefter startar AMS applikationen genom att anropa startApp(). MIDleten hamnar då i aktivt tillstånd. Från detta tillstånd kan MIDleten pausas genom anrop till pauseApp(). AMS kan till exempel pausa en MIDlet så att användaren kan ta emot ett inkommande samtal. När väl en applikation har startats kan man skifta mellan dessa två tillstånd hur många gånger som helst innan AMS eller ett undantag i programmet försätter MIDleten i tillståndet förstörd. Då har livscykeln nått sitt slut och programmet avslutas och ”garbage collection” utförs. [31]



Figur 7 Livscykeln för MIDlets

En MIDlet är alltså den typ av program som körs under MIDP. Allt som ingår i en MIDlet, det vill säga att källkod och bilder packas i en jar-fil (Java Archive). Det kan även vara så att ett antal program packas i en enda jar-fil, då bildar de en MIDlet-suite. Det som är karaktäristiskt för

en MIDlet-suite är att alla ingående program körs i samma sandlåda (Sandbox). Detta gör att MIDletarna kan komma åt varandras publika metoder och dela data mellan varandra i RMS. En jar-fil innehåller även ett manifest som beskriver MIDlet-suiten. För varje .jar-fil finns en motsvarande .jad-fil. Denna används av AMS:en för att hantera MIDleten och den används även av MIDleten själv för att hämta ut konfigurationsattribut. Genom att avläsa attributen i jad-filen kan AMS:en avgöra om MIDleten passar för enheten och man undviker på så sätt onödig nerladdning av den betydligt större jar-filen. Jad-filen måste innehålla dessa attribut:

- MIDlet-Name
- MIDlet-Version
- MIDlet-Vendor
- MIDlet-Jar-URL
- MIDlet-Jar-Size

Följande attribut är inte obligatoriska:

- MIDlet-Description
- MIDlet-Icon
- MIDlet-Info-URL
- MIDlet-Data-Size
- MIDletspecifika attribut som inte börjar med MIDlet-. Exempelvis värden som används för initiering eller värden som är av stor betydelse för alla, i suiten ingående program.

En eller flera MIDlets packas i en .jar-fil som innehåller:

- Ett manifest som beskriver innehållet
- Java klassfiler för en eller flera MIDletar
- Resursfiler (till exempel bilder, animationer och ljud)

Manifestet definierar attribut som används av AMS:en för att identifiera och installera MIDlet-suiten. De flesta av attributen listas i både jar- och jad-filen. Om det i manifestet finns attribut som ej återfinns i .jad-filen används dessa som förvalt värde (default-attribut). [30]

2.13.1 MIDlet-attribut

Namn	**Beskrivning**
MicroEdition-Configuration	J2ME-konfiguration som krävs för att köra MIDleten. Exempelvis

	"CLDC-1.0".
MicroEdition-Profile	J2ME-profil som krävs för att köra MIDleten. Exempelvis "MIDP-1.0".

MIDlet-<n>	För varje MIDlet i en suite skall detta attributet finnas. Värdet är namnet, ikonen, klassen till den n:te MIDleten i .jar-filen separerade med kommatecken. Ex:
	`MIDlet-1: Fotboll,/images/fotboll.png, FotbollMIDlet`
	`MIDlet-2: Rally, /images/rally.png, RallyMIDlet`

MIDlet-Data-Size	De minimikrav i bytes MIDleten har gällande permanent lagringsutrymme. Förvalt värde är 0.
MIDlet-Description	Beskrivning av suiten.
MIDlet-Icon	Namnet på en .png-fil som finns i .jar-filen. Denna bild används som ikon och visas av AMS:en vid valet av MIDlet.
MIDlet-Info-URL	En sökväg till information som i detalj beskriver suiten.
MIDlet-Jar-Size	Storleken på .jar-filen i bytes.
MIDlet-Jar-URL	Sökvägen till .jar-filen.
MIDlet-Name	Namnet på MIDlet-suiten.
MIDlet-Vendor	Den säljare som tillhandahåller suiten.
MIDlet-Version	Versionsnumret på MIDlet-suiten. Formatet är major.minor.micro som i JDK Product Versioning Specification. [50]

JAR Manifestet måste innehålla dessa attribut

- MIDlet-Name
- MIDlet-Version
- MIDlet-Vendor
- MIDlet-<n>
- MicroEdition-Profile
- MicroEdition-Configuration

Dessa attribut är ej nödvändiga i manifestet.

- MIDlet-Description
- MIDlet-Icon
- MIDlet-Info-URL
- MIDlet-Data-Size

[30]

2.14 Klassfiler

Alla klassfiler som används av MIDletarna placeras i .jar-filen utefter en standardstruktur. Punkter i sökvägar ersätts av snedstreck (/). Klassen applikation.funktioner.databas.Connect.class skrivs i .jar-filen som applikation/funktioner/databas/Connect.class. [13]

2.15 Record Management System (RMS)

Trådlösa enheter har normalt två typer av minne, ett för att köra applikationer och ett för beständig lagring. Med beständig lagring menas att data skall finnas kvar även om enheten startas om, batteriet byts eller liknande normal användning. En ordinär databas är för minneskrävande för enheter som använder sig av MIDP. Därför definierar MIDP1.0 en enkel modell för lagring kallad RMS. Varje plattform implementerar RMS på olika sätt, men egenskaperna skall följa MIDP-specifikationen. Varje tabell i RMS kallas ”recordstore” och varje post i tabellen kallas ”record”. Det API som används för att skapa, få tillgång till och ändra posterna beskrivs i javax.microedition.rms. Detta paket definierar en klass; Recordstore och fyra gränssnitt; RecordComparator, RecordFilter, RecordEnumeration och RecordListener. I RMS är varje ”recordstore” en fil som simulerar en tabell i en databas. Varje post i tabellen innehåller ett unikt nummer (recordID) som primärnyckel. Datan lagras i ett fält (array) av bytes. Storleken på lagringsutrymmet som RMS har tillgång till varierar stort mellan enheter. [13][57]

2.16 JavaTM Technology for the Wireless Industry (JTWI)

Under de två senaste åren har flera J2ME-relaterade JSR:er initierats och fokus har då varit på trådlösa enheter. Dessa JSR:er är bland annat MIDP (JSR 37), MIDP 2.0 (JSR 118) och Mobile Media API (JSR 135). MIDP ses som en kommersiell framgång, i och med att MIDP integrerats i en mängd olika produkter. Den växande skaran av JSR:er skapar ett behov av samordning. JTWI har startats för att bland annat beskriva den allmänna arkitekturen och en rekommenderad kombination av tekniker som använder J2ME. JTWI skall också utarbeta en handlingsplan för den riktning utvecklingen av tekniken skall ta. Arbetet i JTWI är alltså tänkt att fungera som en samordnare av alla MIDP-relaterade JSR:er. [25]

2.17 Utveckling av MIDlets

2.17.1 Utveckling

Vid utveckling av MIDlets gås en rad faser igenom. Först måste en idé finnas för en användbar och/eller rolig applikation. Idén måste gå att realisera under de förutsättningar som CLDC/MIDP ger. Realiseringen sker med kod skriven i Java som använder sig av MIDP:s API:er. När källkoden till applikationen är färdigskriven skall koden kompileras. Varje klass i källkoden ger då upphov till en klassfil med bytekod. Förutom kompilering måste filerna genomgå det som i säkerhetsavsnittet beskrivs som pre-verification. Testning följer sedan som ett naturligt steg. Fungerar applikationen som förväntat är det dags att paketera. Vissa applikationer är uppbyggda så att flera MIDlets skall dela data i någon form, vilka då paketeras i en och samma programsvit. De applikationer som ej delar gemensam data packas separat. Efter packning testas applikationen återigen. [30]

Att utveckla MIDlets liknar sättet att utveckla andra program i Java. Klasser skapas som i vissa fall ärver av andra klasser. Metoder modifieras eller läggs till och det önskade resultatet växer fram med tiden. Den stora skillnaden ligger inom testning och felsökning, som är mer krävande än för exempelvis J2SE. För att spara plats och minnesrymd kan utvecklaren i vissa fall vara tvungen att bortse från objektorientering även om det strider mot god programmeringssed. [3][30]

För att kompilera, avlusa, testa, verifiera och paketera MIDlet:s finns det hjälpmedel i form av olika utvecklingsverktyg. Sun har tagit fram

Wireless Toolkit (WTK), som är en uppsättning verktyg för ovanstående uppgifter. WTK är baserat på referensimplementationerna av MIDP och CLDC. WTK går att använda ihop med utvecklingsmiljöerna Sun ONE Studio, Borland JBuilder MobileSet, Metrowerks CodeWarrior Wireless Studio, Oracle9i Jdeveloper med flera. [3][40][41][43]

Med WTK går det att simulera minnesstorlek, heapstorlek, vissa grafikegenskaper, hastighet på den virtuella maskinen och hastighet på nätverksuppkoppling. Dessa egenskaper kan vara viktiga vid testning innan utvecklarna har möjlighet att testa på målenheten. Med WTK följer även ett antal emulatorer. En emulator är ett program som efterliknar hur applikationen kommer att bete sig när applikationen exekverar på målenheten. En stor del av testningen görs i emulatorer eftersom det inte krävs att applikationen laddas ner till målenheten mellan varje ändring i programkoden. Då det i emulatorer inte går att simulera alla egenskaper hos nätverk och målenheter bör utvecklaren ändå överföra applikationen till målenheten. Speciellt viktiga är tester på målenheten när större ändringar gjorts. [3][51]

Till sist skall applikationen distribueras. Distribution kan antingen ske via en operatörs portal eller via en öppen sida på internet.

2.17.2 Nerladdning

Nerladdningen av MIDlets kommer i de flesta fall att ske ”over the air”. Sun fann det därför nödvändigt med ett tillägg till MIDP1.0 och tog fram en specifikation för nerladdning. Over the Air (OTA) User Initiated Provisioning Recommended Practice specificerar hur spridningen av applikationer från server till den mobila enheten bör gå till. Specifikationen är till största delen rekommendationer för egenskaper hos terminaler, nätverksprotokoll och mjukvara. Det är AMS, i vissa sammanhang kallad Java Application Manager (JMS), som har ansvaret för nerladdning, installation och exekvering av MIDleten på enheten. [30][46]

För att kunna hitta applikationer behöver telefonen antingen en WAP-läsare eller ett specialanpassat program. Ett sådant program kan exempelvis öppna mot en portal genom ett enkelt val i telefonens menysystem. Från portalen ges användaren möjlighet att ladda ner MIDlets på ett enkelt sätt. [30][35]

I de flesta fall kommunicerar telefonen med en server via en WAP-gateway. För att det skall fungera måste gateway:en släppa igenom MIME-typerna jar och jad. Före en installation kan påbörjas skall enheten ladda ner en jadfil. Jadfilen innehåller information om MIDleten, exempelvis storlek och version på jarfilen, dess url och försäljare. Baserat på information från jadfilen bestämmer AMS:en om applikationen är kompatibel med enheten. Om applikationen anses vara kompatibel, kontrollerar AMS:en om applikationen finns installerad på enheten. Om inte laddas jarfilen ner. Finns det en version installerad, jämför AMS:en versionsnumreringen i jadfilen med den installerade applikationen. Om versionsnumret är högst i jadfilen betraktar AMS:en det som en uppdatering av applikationen och påbörjar nerladdningen av den nya jarfilen. AMS:en jämför enbart applikationens namn och versionsnummer för att avgöra om nerladdning skall ske. AMS:en ställer inga krav på att uppgraderingen skall komma från samma server som den tidigare versionen. Med andra ord finns risken att en förmodad uppgradering inte alls motsvarar den ursprungliga försäljarens intentioner. Den sista kontrollen som utförs är att informationen i jarfilens manifest jämförs med den som finns i jadfilen. Om informationen är lika slutförs installationen, annars avbryts den av AMS:en. [30][46]

2.18 Alternativa tekniker

J2ME är inte den enda teknik som möjliggör nerladdning av applikationer för exekvering på mobiltelefoner. I uppsatsen har det valts att titta lite extra på svenska Mophun, amerikanska BREW och franska In-Fusio. De är på sätt och vis konkurrenter till J2ME men de kan också samexistera på samma enhet och därmed vara ett komplement.

2.18.1 Mophun

Mophun är en mjukvarubaserad spelkonsol för mobila terminaler. Företaget bakom Mophun är Stockholmsbaserade Synergetix. På Mophunplattformen kallas applikationerna (spelen) gamelets. En gamelet är skriven i C/C++ eller assembler samt Java på experimentstadium. Koden exekveras på en virtuell maskin, så utveckling för olika operativsystem behöver inte ske. Mophun är implementerad för att dra största nytta av värdoperativsystemets grafiska möjligheter. Därför har plattformen kallats mobilvärldens Playstation2, med den skillnaden att det handlar om mjukvara istället för hårdvara. Utvecklingsmiljön (Software Development Kit (SDK)) för

Mophun är gratis och Synergetix plan är att tjäna pengar på att sälja licenser för plattformen. I dagsläget finns mophunplattformen endast på SonyEricssons T300, men Synergetix förhandlar med andra leverantörer av handhållna enheter om att sälja tekniken. Mophun är i dagsläget endast inriktad på spelmarknaden men plattformen kan mycket väl lämpa sig för andra applikationer som drar nytta av dess grafiska specialinriktning. [3][7][40][55]

Så här går det till när en idé skall bli spel på Mophun:

1. För att vara säker på att en spelidé skall kunna bli certifierad, bör den först bli godkänd av Synergetix. Detta steg är dock inte obligatoriskt. Ett avtal mellan utvecklare och Synergetix skrivs innan arbetet med spelet startas. [55]

2. När spelet är färdigutvecklat skickas det till Synergetix för certifieringskontroll.
    - Koden kontrolleras för att säkerställa att den inte är skadlig eller osäker.
    - Spelet testas så att det håller den standard som överenskommits.
    - Spelet får digital signatur som betyder att det är godkänt och att det går att köra på en enhet som har mophunplattform. Den digitala signaturen innebär att inget får ändras såvida inte processen med signering genomförs på nytt. [55]

3. Uppfyller spelet alla kraven certifieras det och skickas tillbaka till utvecklaren. Spelet görs tillgängligt för beskådning av operatörer. [55]

4. Spelet är därmed färdigt för att distribueras och läggas upp på operatörers nerladdningsplatser. [55]

5. Vanligtvis laddas spelen ner över mobilnäten (OTA).
    - Genom exempelvis en wap-portal kan användare ladda ner en så kallad laddningsapplikation. Med denna kan materialet som finns på portalen bläddras igenom och önskat spel väljas. Spelet levereras via WAP med hjälp av push-teknik eller via MMS. Nerladdning är också möjlig via WAP eller som MIDP.

- Innan spelet laddas ner måste distributören med hjälp av ett speciellt verktyg, kallat Vendor Signing Tool, signera spelet för att det skall fungera på mottagarens telefon. Signeringen genomförs för att hindra piratkopiering av spel. Signeringen gör att spelet låses till den specifika telefon som det laddas ner till.
- Slutanvändaren debiteras av distributören när nerladdningen genomförts. Distributören betalar licensavgift för spelet till Synergetix. Har nerladdning gjorts från Mophun:s spelportal betalar Synergetix licenspengar till utvecklarna, annars betalas utvecklarna av distributören. Loggfiler från Vendor Signing Tool hos distributören ligger till grund för betalning. [55]

2.18.2 Binary Runtime Environment for Wireless (BREW)

BREW är en plattform för att köra program på mobiltelefoner med Qualcomms chipset för Code Division Multiple Access (CDMA). CDMA är en teknik för mobiltelefonnät, mest använd i USA och Asien. Även om BREW är framtaget för Qualcomms chipset, så går BREW att applicera på telefoner för andra system, exempelvis GSM. För att skriva applikationer för BREW används normalt C eller C++. Till skillnad från J2ME och Mophun används inte en virtuell maskin som mellanlager mellan kod och hårdvara. Det finns dock möjlighet att skriva i andra språk, exempelvis Java, men då krävs en virtuell maskin. Utomstående företag som till exempel IBM och Insignia förser Qualcomm med tekniken för att kunna köra MIDlets. Ett program som fungerar på en telefon som stödjer BREW går att köra på alla telefoner som har BREW. Strukturen gör det enkelt att utveckla mot BREW, men möjligheten att dra nytta av de mest högteknologiska telefonernas egenskaper försvinner. Lägstanivån sätts av specifikationen för BREW. BREW är utvecklat för att köras på mobiltelefoner vilket speglas i API:t. Det finns bland annat funktioner för att komma åt telefonbok och det finns möjlighet att skriva program där användaren tillåts ringa med hjälp applikationen. [37]

För att skriva och lansera program för BREW bör utvecklarna enligt Qualcomm bli medlemmar i Developer Alliance Program genom vilket de i sin tur får tillgång till ett antal tjänster. Vissa av dessa tjänster kostar pengar och andra kräver att koden är kompilerad under ett visst operativsystem. Ett SDK för BREW skall laddas ner och Microsoft Visual Studio är den rekommenderade utvecklingsmiljön. Vidare skall koden kompileras med en kompilator från RealView. Dessutom skall

utvecklarna bli verifierade av Qualcomm, vilket öppnar upp för tillgång till BREW:s utvecklingsverktyg. vilka är kritiska för att programmet senare skall kunna lämnas in för testning. Därefter kan själva programutvecklingen börja. När applikationen är klar skall den testas för kompatibilitet av Qualcomm. När Qualcomm stämplar programmet som kompatibelt blir det klart för marknaden under förutsättning att det finns en etablerad affärsrelation mellan Qualcomm och utvecklarna. Fördelningen av intäkterna när applikationen väl säljs till konsument är: 80 % till utvecklare, 10 % till nätverksföretaget och 10 % till Qualcomm. [38][39]

2.18.3 In-Fusio

In-Fusio är ett franskt företag med en J2ME-baserad plattform som är optimerad för grafik. Språket är Java, men API:erna är speciella – liknande i-appli:s från NTT DoCoMo. In-Fusio:s plattform heter ExEn vilket kommer av Execution Engine. För att ExEn skall kunna köras krävs att både operatören och telefonen stödjer ExEn plattformen. [8]

För att utveckla mot ExEn behövs In-Fusio:s SDK, Sun:s JDK samt en editor. Därefter kan programutvecklingen starta. För att spelet skall nå marknaden måste det skickas in till In-Fusio som kontaktar utvecklaren om spelet väljs ut för lansering. Blir spelet utvalt får utvecklaren skriva ett kontrakt med In-Fusio för att bli en Premium Developer. In-Fusio garanterar i kontraktet att spelet skall marknadsföras mot de anslutna operatörerna. Om en eller flera operatörer fattar tycke för spelet, testar och justerar In-Fusio spelet tillsammans med utvecklaren. När operatören och de andra parterna är nöjda, läggs spelet upp på In-Fusio:s server för nerladdning genom operatörens nätverk. [9]

2.19 Java i Japan

NTT DoCoMo är den ledande telefonoperatören för mobil kommunikation i Japan. Inom multimedia har DoCoMo en affärsidé, kallad i-mode. I-mode är en mobiltelefonplattform som ger kunder tillgång till en mängd multimediatjänster. I en vidareutveckling av i-mode har DoCoMo tillsammans med Sun utvecklat ”i-appli” (internet applications). I-appli ger kunderna tillgång till avancerade Javabaserade tjänster. DoCoMo:s abonnenter är hänvisade till DoCoMo:s specialanpassade telefoner. Telefonerna är vid inköpstillfället konfigurerade för i-appli, vilket gör det enkelt för kunden att genast börja konsumera tjänsterna. Via DoCoMo:s i-meny

görs tjänsterna tillgängliga för kunderna. För att upprätthålla kvalitativa tjänster har en kvalitetskontroll utformats av DoCoMo.
I-appli tillhandahåller två typer av applikationer:

- Stand-Alone Applications. Applikationer som laddas ner och lagras. Efter nerladdning krävs ingen uppkoppling. Populära applikationer av denna sort är karaoke och spel.
- Agent Type Applications. Applikationer som kräver uppdatering av data vilket möjliggör aktuella aktiekurser och väderleksrapporter.

DoCoMo var redan före utvecklingen av i-appli, duktiga på att skapa ett rikt tjänsteutbud. Tekniken liknade WAP, men då denna typ av gränssnitt är relativt statiskt, blev upplevelsen för användaren inget utöver det vanliga. Med lanseringen av i-appli tillkom möjligheten att tillhandahålla tjänster med avancerade grafiska gränssnitt vilket ökade interaktiviteten dramatiskt. DoCoMo:s telefoner stödjer Secure Sockets Layer (SSL), det vill säga ”end-to-end” kryptering. Detta skapar ökat intresse för utförande av banktjänster och e-post genom mobilen. DoCoMo sålde under första veckan efter lanseringen av i-appli 230 000 tjänster, vilket har väckt ett intresse hos en mängd innehållsleverantörer som till exempel Disney, Sega och Namco.
En anledning till innehållsleverantörernas stora intresse är DoCoMo:s fördelningsmodell av intäkterna från i-appli, som är en del av i-mode. 1 % av i-modes datatjänster har DoCoMo själva utvecklat. 9 % av datatjänsterna utvecklas ihop med officiella innehållsleverantörer, vilka anses erbjuda kvalitativa tjänster. Med leverantörerna skrivs ett speciellt avtal för att tjänsterna skall exponeras. De utvalda leverantörerna ges tillgång till vissa av DoCoMo:s nätverkstjänster. 90 % av tjänsterna ligger på internet och är endast verifierade av DoCoMo. Av det kunden betalar för applikationerna/tjänsterna går 10 % av intäkterna till DoCoMo och resterande del går till utvecklarna/innehållsleverantörerna. Enligt Northstream kan succén för i-mode förklaras med ett nära samarbete mellan DoCoMo och innehållsleverantörerna vilket ger en slags synergieffekt. Northstream identifierar följande faktorer i DoCoMo:s strategi:

- Innehåll och innehållsleverantörer prioriteras högt.
- En attraktiv förtjänstfördelningsmodell.
- Högpresterande terminaler.
- SMS-tjänsten uppgraderades till e-post.

[34][42]

3 Metod

Uppsatsen skall utröna hur situationen ser ut och vilka trender som finns i branschen. När vi valde metod för uppsatsarbetet valde vi mellan ett kvalitativt och ett traditionellt perspektiv. För denna uppsats ansåg vi att det kvalitativa synsättet passade bäst eftersom den förestundande studien antogs innefatta ett fåtal intervjuer med personer besittande stor ämneskunskap.

> Det kvalitativa synsättet eller ”filosofin” riktar intresset mera mot individen. Istället för att fråga hur en objektiv verklighet ser ut ställer man frågan hur individen tolkar och formar en (sin) verklighet.[1]

Det kvalitativa perspektivet är ofta förenat med en induktiv ansats. Detta innebär att man inte utgår från en hypotes. Under datainsamlingen och empirin formulerar man hypoteser eller teorier. Ansatsen innebar att branschens förhållningssätt till problemområdet studerades, utifrån detta kunde en adekvat slutsats växa fram, om var branschen befinner sig och hur den antas utvecklas. [1]

I detta avsnitt beskrivs vilket tillvägagångssätt som använts. Arbetet har bedrivits i tre steg:

1. Litteraturstudier och praktisk övning i programmering mot J2ME.

2. Genomförande av djupintervjuer med, i branschen väl insatta personer.

3. Analys av intervjuer och analys av det praktiska arbetet med programutveckling.

I de följande styckena i detta kapitel förklaras det i detalj vad som menas med de ovanstående delmomenten.

3.1 Litteraturstudier och programmering

Litteraturstudier har haft en central roll i att ge författarna en teoretisk ram kring problemområdet. Behovet av litteratur har varit stort för att överhuvudtaget kunna genomföra såväl intervjuer som praktiskt programmeringsarbete. Då författarna inte hade någon egentlig uppfattning av vad J2ME var togs det snabbt beslut om att sätta sig in i hur och när J2ME kan användas. Detta gjordes genom litteraturstudier och studier av tidigare forskning samt praktisk övning i färdigheten att programmera för J2ME med inriktning mot

mobiltelefoner. För att undvika snedvriden information, söktes den från flera, av varandra oberoende källor.

Programmeringsfasen gav upphov till en uppsjö av frågeställningar. Dessa låg till grund för ett antal frågor vilka skulle besvaras under intervjufasen. Det inledande arbetet med litteraturstudier och programmering gav god insikt i vilka grupper av företag som är intresserade av Java för mobiltelefoner. Intressenterna är främst nätverksoperatörer, telefontillverkare samt de som skriver programvara för telefoner och servrar.

3.2 Intervjuer

Som primär datainsamlingsmetod har det för uppsatsen valts intervjuer, då det i litteraturen finns stora luckor kring problemområdet.

3.2.1 Intervjumetod

I uppsatsen har det valts att använda delvis strukturerade intervjuer, med färdigformulerade frågor, men med möjlighet för respondenten att svara öppet på frågorna.

> I en delvis strukturerad intervju vill man ha viss information från alla respondenter. Dessa intervjuer styrs av ett antal frågor eller frågeställningar som skall utforskas, men varken ordalydelse eller ordningsföljd bestäms i förväg. Detta slags intervju gör det möjligt för forskaren att svara an på situationen som den utvecklas, på respondentens bild av världen och på nya idéer som dyker upp. [27]

Intervjuerna låg till grund för att få en förståelse om problemområdet samt som faktabank vid analys av resultatet.

3.2.2 Urval av informanter

Urvalet av undersökningspersoner är en viktig del av undersökningen, och personerna skall väljas med omsorg. De aktörer på hemmamarknaden som ansågs vara relevanta, och de med störst inflytande över denna typ av tjänster valdes ut för kontakt. Det kändes viktigt för författarna att få en helhetsbild och därför valdes flera segment ut för intervjuer. Dessa var nätverksoperatörer, telefontillverkare och tjänsteleverantörer. Ansatsen för uppsatsen var

att ta reda på vart branschen generellt sätt är på väg. För att inte beskriva något specifikt företags strategier har det, i den mån det varit möjligt, gjorts intervjuer med personer från olika företag i varje segment. Vid efterforskningar kommit fram till att det finns ett fåtal personer på ett litet antal företag som är väl insatta i problemområdet. De personerna vi intervjuat har befattningarna: Business Developer Manager på Sun Microsystems AB, Java Product Manager på Sony Ericsson Mobile Communications, Research & Development Manager på Mediabricks AB, Verkställande direktör på Gnistra AB, Business Developer på Halebop AB och Produktchef för mobiltelefoni på Tele2. [6]

3.2.3 Genomförande av intervjuer

För att ge intervjupersonerna möjlighet att förbereda sig på bästa sätt skickades frågorna ut till dem via e-post. Därpå bokades tider för intervjuer med dem. Vi uppskattade att intervjuerna i genomsnitt skulle pågå i 45 minuter så vi bad respondenterna avsätta cirka 1 timma för intervjuerna. För att få ut så mycket som möjligt av intervjuerna genomfördes de genom trepartssamtal i telefon eller som personligt möte. I ett fall förekom brevväxling via e-post som substitut till intervju. Vid både den personliga intervjun och telefonintervjuerna användes en digital bandspelare för inspelning av samtalen. Bandspelaren bidrog till att intervjuerna var lätta att genomföra då heltäckande anteckningar inte samtidigt behövdes tas under intervjun. Vi är medvetna om att respondenterna kan känna sig lite obekväm med vetskapen om att samtalet spelas in. Till fördelarna kan tilläggas tryggheten i att ha respondenternas exakta svar inspelade vid analys av resultat. För att kunna arbeta effektivt i analysfasen anser vi att de inspelade intervjuerna måste transkriberas till sökbar text, vilket är ett tidskrävande arbete. Vår uppfattning är dock att fördelarna med inspelning gott och väl uppväger nackdelarna.

3.3 Analys av intervjuer och praktiskt arbete

Då respondenterna i intervjuerna gavs tillfälle att prata ganska fritt kring de frågor vi ställde till dem, fick vi ett omfattande material att analysera. För att se mönster gjordes en grov sammanställning av det insamlade materialet. Erfarenheter från litteraturstudier och eget arbete vävdes in i en första, grov analys där vi letade efter samband och meningsskiljaktigheter mellan respondenternas svar. Område för område av problemformuleringen analyserades därpå, vilket resulterade i förfinade resultat. [4]

3.4 Validitet och reliabilitet

Med validitet menas en mätmetods förmåga att mäta det man avser att den skall mäta. Det finns två typer av validitet, intern och extern. Intern validitet avser trovärdigheten på materialet. För att kunna garantera en hög grad av validitet skickades intervjufrågorna ut till respondenterna via e-post i förväg. Därefter lämnades en utskrift av materialet till intervjupersonerna så att eventuella missförstånd av det insamlade materialet kunde undvikas. För att belysa problemområdet så tydligt som möjligt ville vi beskriva det ur olika synvinklar, så kallad källtriangulering. Vi valde därför ut personer med olika relationer till problemområdet. Det som menas med en extern validitet är att det inte är författaren som definierar generaliserbarheten. Läsaren får ta del av materialen och de slutsatser som ges och får sedan själv avgöra generaliserbarheten. [4]

Med reliabilitet menas att den kunskap som kommer fram är framtagen på ett tillförlitligt sätt. Inom den kvalitativa ansatsen används både teknisk utrustning och människan som mätinstrument. För att kunna dra relevanta slutsatser av intervjumaterialet var det viktigt att få en bra kvalitet på de inspelningar som gjordes. Som nämndes tidigare användes en digital bandspelare. För telefonintervjuerna användes dessutom en ”teletaper” som underlättar inspelning via telefon. Det personliga mötet spelades in med en professionell mikrofon med högkvalitativ ljudåtergivning. För att vara på den säkra sidan fördes det, under intervjuerna stödanteckningar av vad som sades för att kunna återge vad som sagts i det fall ett tekniskt missöde skulle inträffa. [4]

4 Analys av resultat

4.1 Utvecklingen

Att komma igång med programutveckling för MIDP är enkelt. Allt från kodexempel till kompletta utvecklingsmiljöer finns lättillgängligt på internet. Flera av terminaltillverkarna tillhandahåller gratis utvecklingsverktyg och API:er. Carlsson framhäver att det aldrig är bra att bli för beroende av en och samma utvecklingsmiljö. Risken finns att miljön inte utvecklas i samma takt som tekniken. En väg att gå är att skapa sin egen miljö. [3][26]

Vid utvecklingen mot J2ME/MIDP är det viktigt att hålla nere antalet klasser som används, eftersom varje klass tar upp rikligt med minne. ”Vi plockar bort en tredjedel av storleken genom att lägga alltihop i en

enda stor klass” säger Carlsson. Det leder till att det objektorienterade synsättet får läggas åt sidan. Minnesrymden i terminalerna är så pass begränsad att objekt ofta behöver sättas till null för att spara plats. För att öka prestandan hos MIDP-applikationer bör utvecklare vara insatta i hur den virtuella maskinen behandlar datatyper och dess operationer. Stränghantering tas upp som kritiska operationer då de kräver mycket minne vid behandling av den virtuella maskinen. Utvecklarna råder därför till försiktighet med användandet av strängar. [3][26]

Behovet av att komma åt enhetsspecifika metoder (native) ses av utvecklarna som stort. Vissa terminaltillverkare har modellspecifika API:er, vilket är ett steg på vägen till åtkomst av omkringliggande resurser. De modellspecifika API:erna kan tänkas ge tillgång till ljudsystem, vibratorer, ljus, videouppspelning och förbättrade grafikegenskaper. När utveckling sker mot avancerade telefonmodeller med hög kapacitet vill utvecklarna dra nytta av hela deras register av funktioner, och inte låta sig begränsas av en alltför generell MIDP-specifikation. Friman uttrycker sig så här: ”man vill i en kapabel telefon dra nytta av hela dess kapacitet”. API:er för speciella telefonmodeller är ett tydligt uttryck för att standardiseringsprocessen inom JCP går för långsamt. Lundberg hävdar dock: ”Tillverkarspecifika API:er är en nödvändighet för att få en innovationskraft bakom tekniken. Det som är utmaningen sen är att se till att de här innovationerna snabbt sugs in i standarden och kommer i nästa generation”. [3][26][32][41]

Vid klient-server lösningar skall klienten utföra så mycket som möjligt av arbetsbördan. Anledningen till det synsättet är att svarstiderna i telefonnäten är långa. Om klienten är beroende av att ständigt få uppdateringar via nätet, kan svarstiderna hindra en bra användarupplevelse. En lösning kan vara att låta animationer visas i förgrunden för att ge användaren illusionen av att det inte finns fördröjningar i infrastrukturen. Under tiden hämtas data i bakgrunden från en server via nätverket. [3][26]

Tillverkarna ges frihet att implementera MIDP på det sätt de finner lämpligt. Utvecklarna stöter ofta på problem som grundar sig i hur tillverkarna löst implementationen av vissa funktioner. Skillnaderna ger upphov till att applikationerna måste anpassas till i stort sett varje enhet eller modellserie. En stor del av utvecklingscykeln består av testning och anpassning för de olika enheterna. Många av testerna utförs i emulatorer, men viktigare är att testa på målenheten. Det

räcker inte med att applikationen fungerar på terminalerna, programmet skall kunna laddas ner OTA för att anses användbart. Trots att det inte finns en övre gräns för storlek på MIDlets, bör storleken begränsas till 30 kilobytes. Ibland finns begränsningen i terminalen, men oftare finns den i nätverket. [3][26]

4.1.1 Diskussion och slutsats:

Flera av de utvecklingsmiljöer som finns att tillgå är lätthanterliga och ger ett gott stöd till utvecklaren. En bas för flera utvecklare är Sun:s WTK. För att inte bli beroende av en enskild utvecklingsmiljö och för att kunna ligga i utvecklingens framkant använder sig en del programmerare av egenframtagna verktyg.

J2ME/MIDP har hamnat ganska långt ifrån Javas slogan ”Write once, run anyware”. Vid en första anblick verkar det precis så lätt som ”Write once”, men vid närmare granskning tornar problemen upp sig. Tillverkarna har speciella API:er och utvecklarna kan inte lita på att liknande saker utförs likadant på olika enheter.

Säkerhetsmodellen i CLDC/MIDP medför trygghet för utvecklare eftersom deras applikationer inte kan förstöra omgivningen i telefonen. Samtidigt innebär säkerhetsmodellen ett problem eftersom programmerarens möjligheter begränsas betydligt.

För att MIDP skall tillfredställa utvecklarskaran måste standardiseringstakten öka så att MIDP-NG inte är förlegad vid lansering. Utvecklingen ligger alltid före en standardisering och det kommer därför alltid att finnas behov för enhetsspecifika API:er. JCP har en viktig roll i att uppta godbitarna ur dessa API:er i standarden så snabbt som möjligt för att undvika fragmentering.

4.2 Möjligheter

J2ME/MIDP är en relativt ny plattform som fortfarande växer. För att MIDP inte skall drabbas av växtvärk krävs en rad saker.
Först och främst måste tekniken gynnas av operatörerna. Alla är beroende av operatörerna och om de tvekar stannar hela branschen upp och förtjänst uteblir. Operatörerna är försiktiga efter tidigare lanseringar av nya tekniker som inte slagit väl ut. Lars Lundberg säger: ”Wap är ju det skinande exemplet. Någonting som finns i 99% av gsm-nallarna men som används av 4%.” För att Java inte skall gå i WAP:s fotspår, krävs det snarare en tjänstelansering än en

tekniklansering. Tjänsterna måste dessutom vara attraktiva och lätta att använda. När tjänsterna lanseras är det viktigt att betalningsmodellerna passar kunderna och att förtjänstdelningen passar utvecklare och operatörer. Möjligheten för kunden att se demonstrationer eller testa produkterna före köp, tros vara en viktig del för att få kunden att känna sig trygg i sitt val. För att få en motsvarande trygghet hos den som säljer innehållet, är kravet på säker hantering av rättigheter (Digital Rights Management (DRM)) viktigt. Fungerar inte säkerheten vågar inte innehållsleverantörerna lansera applikationer och riskera sina investeringar. [3][5][32][41][40]

Begränsningarna ligger inte bara hos operatörerna utan även hos terminaltillverkarna.

> Det krävs en större spridning av terminaler som kan spela javaapplikationer på ett tillräckligt bra sätt och tillräckligt attraktivt, det vill säga, inte svartvita skärmar, inte för långsamma, inte för lite minne, det är en väldigt viktig faktor, tror jag. [56]

Kraven på bättre terminaler måste uppnås för att attrahera konsumenterna och ge dem en förväntad upplevelse eller bättre. [3][26][32][56]

MIDP2.0 ger utökat stöd för bland annat grafik, kryptering, pushteknik, ljud och för att sätta tillståndsnivåer för känsliga operationer. Grafikegenskaperna i MIDP2.0 tillåter utvecklare att skriva intressantare applikationer, i synnerhet spel. Pushteknik ger både spel och nyttoapplikationer möjlighet till flexiblare tjänster. Nätverksspelen spås en lysande framtid på marknaden med pushtekniken som en möjliggörande faktor. Med kryptering kommer det finnas en stor potential för affärskritiska applikationer att utvecklas och frodas. En marknad öppnas för mobila applikationer som en del i ett större affärssystem. Användaren kan ge en MIDlet-suite tillstånd att utföra operationer som anses känsliga, exempelvis uppkoppling mot nätverk, vilket kostar pengar.
[3][18][32][40]

4.2.1 Diskussion och slutsats

J2ME har i dagsläget ett enormt industristöd. Både terminaltillverkare, mjukvaruföretag, nätverksoperatörer och leverantörer av infrastruktur samarbetar i JCP för att få fram standarder. Antalet terminaler som har stöd för Java ökar hastigt och i samma takt kommer efterfrågan på

MIDlets att öka. Java har en stor utvecklarkår som borde kunna leverera program i den utsträckning som krävs. När nätverken uppgraderas och latenstiderna blir kortare kommer användning av nätverksspel ta fart på allvar. [3][26][32][40][41]

De konkurrerande teknikerna BREW, ExEn och Mophun kanske kommer att samexistera med MIDP, om det finns ett behov av bättre grafikstöd och tillgång till operativsystemnära funktioner som inte MIDP stödjer. Moser anser att Mohpun ser intressant ut och möjligen kan ersätta J2ME för spel. [5][40][41]

4.3 Tekniken

Med WAP fanns intentionen att det skulle vara metoden för hur allt innehåll till mobilen skulle laddas ner. Det inkluderade ett wml-skript som tillät minimala applikationer att exekveras på telefonen. WAP fick emellertid inte det genomslag som man hoppades på. Sun var tidigt ute med J2ME som lämpade sig för exekvering på resursbegränsade terminaler, där säkerhet och portabilitet är viktiga faktorer. Om Java för mobiltelefoner säger Häger ”Det är en bra metod och teknologi för att lösa det som behöver lösas. Sun har lyckats få med sig alla på det momentet”. [28][40]

Anledningen till att J2ME fått stort genomslag är att det finns tillgängligt för terminaltillverkarna. Istället för att skapa en egen teknik licensierar de en ”de facto” standard.

I JCP har MIDP utvecklats sedan 1999 och den första versionen lanserades 19 september år 2000. Förslaget till specifikationen kom från Motorola och före det hade Sun genom JCP påbörjat arbetet med CLDC. [12][13]

Sun driver utvecklingen av J2ME för att få en närvaro på klientsidan, vilket i sin tur ger en merförsäljning på serversidan, som är Sun:s primära affärsområde. Tekniken för CLDC/MIDP licensieras av terminaltillverkare. Tekniken drivs i ett initialt skede framåt av tillverkare som tillhandahåller enheter. Syftet är främst att skapa en attraktivare produkt och på så sätt locka till sig kunder.

> I Europa så är telefontillverkarnas ställning ganska stark. Så därför har telefontillverkarna väldigt stor möjlighet att påverka vad som händer. Operatörernas ställning är klart sämre i relation till hur operatörernas ställning ser ut i Japan. [56]

I nästa fas kommer förmodligen operatörerna vara den drivande kraften eftersom de genom Java-tekniken får möjlighet att skapa trafikgenererande tjänster. För att inte behöva börja på ruta ett när 3G-näten tas i bruk är det viktigt för operatörerna att skapa nya beteenden hos användarna. [5][34][41][40]

Genom JCP medverkar mjukvaruföretag för att J2ME skall blir en bra plattform för applikationsutvecklare. Blir J2ME attraktivt för utvecklare kommer intressanta produkter att skapas, vilket är nödvändigt om Java-tekniken skall få genomslag på marknaden. [13][18]

4.3.1 Diskussion och slutsats

Anledningen till att Sun framgångsrikt lyckats övertyga stora delar av branschen för trådlös kommunikation är delvis att Sun snabbt insåg behovet av en exekveringsmiljö i trådlösa enheter. En ytterligare anledning är att branschen inte vill ge Microsoft möjlighet att skaffa sig samma ställning på telefonsidan som på persondatorsidan där de nära nog tillskansat sig en monopolställning. Operatörer och terminaltillverkare vågar, tack vare Javas säkerhetssystem, släppa in tredjepartstillverkade program in i sina nät och telefoner. De kan lita på att användarna inte får obehagliga upplevelser på grund av säkerhetsluckor, då Java körs i en skyddad miljö. [3][26][40][41]

4.4 Samarbete mot ett gemensamt mål?

På en hög nivå samarbetar parterna i Java Community Process (JCP). Här får parterna möjlighet att påverka så att specifikationerna stämmer överens med deras intressen. JTWI samordnar utvecklingen av MIDP-relaterade tekniker. Dels för att komma till rätta med fragmenteringen, men också för att presentera riktlinjer för hur berörda tekniker samverkar på bästa sätt. [25][40][41]

En utvecklare som vill sälja en applikation till slutkund måste i de flesta fall samarbeta med en operatör. Operatören har erfarenhet av att marknadsföra produkter och tjänster mot slutanvändare.

> Idag finns det inga riktigt bra andra sätt för en tredjepartare till exempel en utvecklare av ett javaspel att ta betalt av en slutanvändare utan att använda operatörens betalningstjänst. Det finns ju kreditkort men det är alldeles för svårt. [41]

Operatören är den enda som har en utvecklad betalrelation till användarna, vilket gör att samarbetet i dagsläget sker helt på operatörernas villkor. [3][32][41]

Telefontillverkare och utvecklare har i vissa fall ett tätt samarbete. Utvecklare får support från terminaltillverkare, som i sin tur får felrapporter från utvecklare. Samarbetet leder till att terminaler kan vidareutvecklas och att applikationer blir mer anpassade till aktuella telefonmodeller. Framöver kommer telefontillverkare att utöka samarbetet med utvecklare, då Javaapplikationer kommer att förinstalleras i telefonerna i större utsträckning än tidigare. [26][40]

Ett samarbete mellan terminaltillverkare, för att driva fram gemensamma lösningar, skulle gynna många inblandade. Först skulle utvecklarna jubla, då de inte skulle behöva anpassa sina program efter varenda modell på marknaden. Operatörerna skulle glädjas över att slippa tillhandahålla terminalspecifika tjänster och stå för omfattande support. [5][41][56]

Men vem kommer att leverera tjänsterna? I ett initialt skede kommer det nog att vara de stora applikationsleverantörerna som har möjlighet att ta en stor kostnad för att etablera sig i segmentet. De är inte beroende av att just de mobila tjänsterna ger snabb avkastning. När beteendet väl finns och efterfrågan är hög, kommer det att finnas utrymme för utvecklare med applikationer riktade mot ett visst marknadssegment. [3][56]

4.4.1 Diskussion och slutsats

För att J2ME skall få fäste på marknaden krävs att de berörda parterna arbetar mot ett gemensamt mål. Lundberg proklamerar: ”det är faktiskt så att Skandinavien är den sämst utvecklade delen av den mobila världen just nu, vad det gäller javatjänster”. Nyckeln till framgång är en rättvis modell för förtjänstfördelning främst mellan operatörer och innehållsleverantörer. Alla led i värdekedjan måste ges möjlighet att tjäna pengar. Ett framgångsrikt exempel är DoCoMo:s system för förtjänstfördelning. JTWI har en viktig roll i att vägleda de övriga expertgrupperna runt MIDP. Standarden måste bli tydlig och processen tillräckligt snabb för att intresset runt MIDP skall bibehållas. [26][40][41]

4.5 Åt vilket håll är J2ME på väg?

År 2007 spår analysföretaget Zelos group att Java kommer vara den dominerande plattformen inom den trådlösa sektorn. Java kommer att finnas i 450 miljoner terminaler världen över. Motsvarande 74 % av telefonerna som säljs det året spås ha Javastöd. Med den penetrationen av marknaden kommer konkurrerande tekniker inte ha en betydande marknadsandel. Om detta stämmer är svårt att sia om, men i branschen finns en sprudlande optimism inför vad tekniken har för möjligheter. Det finns mycket som talar för att Java blir en de facto standard för exekveringsmiljöer i mobila terminaler. De främsta faktorerna bakom Javas förväntade framfart är; det stora industristödet och den breda utvecklarskara som programmerar i Java. [58]

Om blickarna vänds mot Japan, där Java redan är etablerat som exekveringsmiljö, har spel- och underhållningstjänster haft störst kommersiell framgång. Förhoppningen är att utvecklingen blir liknande i Europa. På den europeiska kontinenten finns det gott om affärsfolk som får sin mobiltelefonräkning betald av arbetsgivaren. Detta segment öppnar upp marknaden för nyttoapplikationer och så kallade infotainmenttjänster. [5][34][41]

Det är tydligt att branschen tror på Java som en plattform för att leverera intressanta tjänster. Flera operatörer har under hösten 2002 lanserat Javatjänster på sina portaler. För att Java skall bli den succé som många hoppas på är det viktigt att det finns ett utbud av kvalitativa tjänster till ett rimligt pris. En förutsättning för det är att modellen för förtjänstfördelning av den förmodade vinsten passar alla parter. [26][32][34][41]

5 Författarnas reflektion

Sedan några år tillbaka har användare haft möjlighet att ladda ner nya ringsignaler och logotyper samt byta skal på sina telefoner. Nu har människans behov av att uttrycka sin personlighet fått ännu en kanal. Turen har kommit till mobiltelefonernas inre i form av nerladdnings-bara applikationer. Vi tror att innehållet i telefonen snart kommer att vara lika viktigt som den yttre ”looken”.

Under processen med uppsatsen har målet varit att nå en slutsats om huruvida Java kommer att vara den exekveringsmiljö som blir standard för denna typ av applikationer eller inte. Med hjälp av prognoser, betraktelser och kommentarer från folk i branschen har vi

under arbetets gång bildat oss uppfattningen att Java kommer att bli de facto standard som exekveringsmiljö på mobiltelefoner.

Java har som tidigare nämnts en stor utvecklarskara. För den som tidigare utvecklat i Java är steget till J2ME kort. Så risken att det finns för få utvecklare tror vi inte finns. Men att börja programmera för J2ME har sina överraskningar i form av strama minneskrav och att programmeraren får lägga objektorienteringen på hyllan i många fall. Det som talar för att många utvecklare kommer satsa J2ME snarare än andra tekniker är förstås antalet terminaler på marknaden, men också att det är så lätt att hitta information på internet och i böcker.

I våra efterforskningar har vi funnit att standarden på tjänster och applikationer för närvarande är väldigt skiftande. I nuläget skrivs många applikationer utan egentlig målgrupp och syfte, vilket delvis beror på att utvecklarna har svårt att se behoven hos användarna. I takt med att fler utvecklare utvecklar tjänster för mobila terminaler kommer kundernas behov att bli tydligare och tjänsterna bättre. I nuläget är det spel som har störst spridning, men marknaden för så kallad infotainment och nyttoapplikationer kommer troligtvis att öka framgent. Helt klart är att det går att skapa intressanta tjänster till telefoner. Det viktiga är bara att lyckas finna tillämpningsområden för tekniken.

Inom en snar framtid kommer ett stort antal terminaler på marknaden ha Javastöd. Operatörer, utvecklare och terminaltillverkare kommer få möjlighet att skapa nya inkomstkällor. Vilken typ av tjänster och applikationer som kommer att få användaracceptans är svårt att sia om men det viktiga för de involverade parterna blir att hitta marknadens behov och kunna nå ut med tjänsterna till de potentiella kunderna.

Det är lätt att måla upp drömscenarion och se mängder möjligheter för J2ME men en sak är säker:

> "Prediction is very difficult, especially if it's about the future."
> -Niels Bohr

Vi tror J2ME är här för att stanna och kanske kan det ge oss applikationer och tjänster som underlättar arbetslivet. Det kan ge oss förströelse när vi väntar på bussen eller på nästa hajdyk…

6 Utvärdering

Inledningsvis, så här i slutet av arbetet, kan vi konstatera att vi känner oss nöjda på de flesta punkter. Det inledande programmeringsfasen gav oss värdefulla insikter och idéer för det fortsatta arbetet med uppsatsen. Denna fas tillsammans med djuplodande efterforskningar var nyckeln till utformningen av intervjufrågorna. I vår jakt efter intervjupersoner sökte vi personer med stor erfarenhet och kunskap om branschen. Dessa personer är få och dessutom väldigt upptagna. Trots detta har vi genom ihärdigt arbete och ett visst mått av tur lyckats genomföra intervjuer med de flesta av de personer vi önskat prata med. Det skall dock sägas att alla vi kontaktat inte varit välvilligt inställda till att intervjuas, dels på grund av sekretesskäl men också av tidsbrist. Om uppsatsarbetet hade förflutit under en längre period och/eller varit sekretessbelagt kunde underlaget för uppsatsen ha blivit mer omfattande.

7 Förslag till fortsatt arbete

Vår uppsats ger en god inblick i hur branschen ser ut och fungerar. Den skildrar även tankegångar om hur tekniken och samarbetet kan utvecklas framöver. För fortsatt arbete och fördjupning anser vi att kontakt med fler företag med en större geografisk spridning krävs. Utvecklingen i närområdet tror vi beror på beslut som fattas av företag som inte har sitt säte i Sverige.

Vi anser att det inom flera, av de i uppsatsen berörda områden, finns möjlighet till vetenskaplig fördjupning. En första utvidgning skulle kunna bestå i en studie över optimering av prestanda i enheter som använder sig av MIDP. Denna studie bör inriktas på objektorientering och minnesåtgång och resultera i rekommendationer hur kod skall skrivas för att exekvera så effektivt som möjligt på resurssnåla enheter som använder sig av J2ME/MIDP.

Ett andra fördjupningsområde är säkerhet. Vår uppfattning är att säkerhet är ett mycket viktigt område för branschen. Att kunna säkerställa att resurser inte utnyttjas av dem som inte innehar rättigheter kräver tillfredställande lösningar från innehållsleverantörer. Från slutkunder kommer det krävas att ”elak kod” inte förstör deras enheter. Forskning inom området bör resultera i säkerhetsmodeller som skyddar såväl slutkund som leverantörer.

Betalningsmodeller har vi belyst som en viktig faktor för att J2ME skall nå framgång. Hur dessa skall se ut för att tillfredställa hela kedjan,

från idémakare till slutkund, ser vi som ett potentiellt forskningsområde.

Vi har undersökt samarbetet i branschen som det ser ut idag. Framtiden kommer med stor sannolikhet att se annorlunda ut. Nya spelare kommer till och andra kanske faller bort. Vår uppfattning är att det för en oinvigd är svårt att greppa hur samarbetet mellan parterna i branschen ser ut. Samarbete är därför ett område som kan behövas ses över även i framtiden.

Under de intervjuer vi genomfört har flera röster höjts om att standardiseringsprocessen går alldeles för långsamt. Alla vill ha en standard att följa, och alla vill att den skall vara attraktiv. Hur kan det åstadkommas för en teknik under snabb utveckling?

8 Förklaringar av akronymer och andra begrepp

AMS	Application Management Software – Kontrollerar en MIDlet för att avgöra om den kan köras på den aktuella enheten. Startar och avslutar applikationen.
API	Application Programming Interface. Kan ses som en verktygslåda med färdiga funktioner och klasser som någon redan har skrivit. I API:t beskrivs vad de utför hur de kan användas.
AWT	Abstract Window Toolkit – Standard API:t för hantering av grafiska användargränssnitt.
CDC	Connected Device Configuration – Konfiguration inom J2ME för enheter som har en stadig uppkoppling mot nätverk och stabil strömförsörjning.
CLDC	Connected Limited Device Configuration – Konfiguration inom J2ME för batteridrivna enheter med tillfällig uppkoppling mot nätverk.
EC	En Executive Committee (EC) representerar ett tvärskikt av intressenter från näringslivet och andra medlemmar ur Java Community. EC är ansvariga för att godkänna och släppa igenom JSR:er. Det finns två olika EC; en för J2SE/J2EE och en för J2ME.
Garbage collection	Systemegenskap som återlämnar minnesblock som inte längre används.
http	HyperText Transfer Protocol – Protokoll för överföring av data. Har använts på internet sedan 1990.
J2ME	Java 2 Micro Edition.
JAD	Java Application Descriptor – Används av AMS för styrning av nerladdningar av program.
JAR	Java Archive är ett plattformsoberoende filformat som lägger samman många filer till en fil.
JSR	En JSR är en Java Specification Request. Det är ett

	dokument som skickas till PMO av en eller flera medlemmar för att föreslå en ny specifikation inom Java. Det finns för närvarande mer än 90 Java specifikationer under utveckling inom programmet för JCP.
JVM	Java(TM) Virtual Machine (JVM) Den del i Java-exekveringsmiljö som är ansvarig för tolkningen av Javas bytekoder.
JVMS	Java Virtual Machine Specification. Specifikation som beskriver hur standardimplementationen av Java Virtual Machine skall fungera. Beskriver även klassfilsformatet.
KVM	Kuaui Virtual Machine eller Kilo Virtual Machine. Specialskriven virtuell maskin för J2ME:s konfiguration CLDC. Storleken ligger beroende på konfiguration mellan 40-80 Kbyte.
MIDP	Mobile Information Device Profile – Profil för mobiltelefoner (CLDC) i J2ME.
PMO	Process Management Office är en grupp inom Sun som är utsedd att övervaka Java Community Progress och sköter de dagliga sysslorna inom programmet.
RAM	Random Access Memory – Minne där minnescellerna är åtkomliga utan att de tidigare cellerna behöver läsas.
RMI	Java Remote Method Invocation – En distribuerad objektmodell för Java-till-Java-program. RMI kan ge tillgång till Java-objekts metoder trots att objekten finns på en annan server.
ROM	Read Only Memory – Minne som endast går att läsa från. Det går ej att uppdatera.

9 Referenser

[1] Backman, J (1998). Rapporter och uppsatser. Lund: Studentlitteratur.

[2]developer.com(2002-12-16). J2ME Core Concepts, http://www.developer.com/java/j2me/print.php/1378971

[3] Gnistra AB (2002-11-22), Intervju med Jacob Friman, Verkställande direktör.

[4] Gunnarsson, Ronny (2003-01-05) Validitet och reliabilitet. http://infovoice.se/fou/

[5] Halebop AB (2002-12-12), Intervju med Jens Moser, Business Developer.

[6] Holme, I.M.,Solvang, B.K. (1991). Forskningsmetodik. Lund: Studentlitteratur

[7] informa media group(2002-12-17). GAMES ANALYST July 12, 2002, http://www.babelmedia.com/pdf/GA9.pdf

[8] IN-FUSIO (2002-12-16). in-fusio - developer zone, http://developer.in-fusio.com/faqs/faq.php

[9] IN-FUSIO (2002-12-16). in-fusio - developer zone, http://developer.in-fusio.com/information_process.php

[10] Java Community Process (2002-10-23). Information Module Profile (JSR 195). http://jcp.org/jsr/detail/195.jsp

[11] Java Community Process (2002-10-17). JSR-000030 J2ME Connected, Limited Device Configuration. http://jcp.org/aboutJava/communityprocess/final/jsr030/index.html

[12] Java Community Process (2002-10-17). JSR-000036 J2METM Connected Device Configuration 1.0a Maintenance Release. http://jcp.org/aboutJava/communityprocess/final/jsr036/index.html

[13] Java Community Process (2002-10-23). JSR-000037 Mobile Information Device Profile (MIDP) (Final Release). http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html

[14] Java Community Process (2002-10-23). JSR-000046 J2METM Foundation Profile Specification 1.0a Maintenance Release. http://jcp.org/aboutJava/communityprocess/final/jsr046/index.html

[15] Java Community Process (2002-10-23). JSR-000062 Personal Profile Specification (Final Release). http://jcp.org/aboutJava/communityprocess/final/jsr062/index.html

[16] Java Community Process (2002-10-23). JSR-000066 RMI Optional Package Specification version 1.0 (Final Release). http://jcp.org/aboutJava/communityprocess/final/jsr066/index.html

[17] Java Community Process (2002-12-15). JSR 75 PDA Profile for the J2METM Platform, http://jcp.org/en/jsr/detail?id=75

[18] Java Community Process (2002-12-19). JSR-000118 Mobile Information Device Profile 2.0 - Final Release, http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html

[19] Java Community Process (2002-12-16). The Java Community Process(SM) Program - JCP Procedures - How the Process Works, http://www.jcp.org/en/procedures/overview

[20] Java Community Process (2002-12-16). The Java Community Process(SM) Program - Introduction - Program Overview, http://www.jcp.org/en/introduction/overview

[21] Java Community Process (2002-12-16). The Java Community, Process(SM) Program - Introduction - Timeline View http://www.jcp.org/en/introduction/timeline

[22] Java Community Process (2002-12-16). The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 82, http://jcp.org/en/jsr/detail?id=82

[23] Java Community Process (2002-12-19). The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 129, http://jcp.org/en/jsr/detail?id=129

[24] Java Community Process (2002-12-19). The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 134, http://jcp.org/en/jsr/detail?id=134

[25] Java Community Process (2002-12-16). The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 185, http://jcp.org/en/jsr/detail?id=185

[26] Mediabricks AB (2002-12-17), Intervju med Magnus Carlsson, Research & Developement Manager.

[27] Merriam, S.B. (1994) Fallstudien som forskningsmetod. Lund: Studentlitteratur.

[28] Micro Java Network (2002-12-11). Micro Java Network - The J2ME Resource : The Battle For BREW, J2ME And Related Technologies, http://www.microjava.com/articles/perspective/shosteck

[29] Mobic AS (2002-12-16). Mobile Games Revenue Set to Increase Ten-Fold in Three Years, Says Analysys, http://www.mobic.com/news/publisher/view.do?id=1503

[30] Muchow, J. W. (2002). Core J2METM Technology & MIDP. Upper Saddle River, NJ: Prentice Hall PTR.

[31] Nokia Corporation (2002-12-16). A brief introduction to MIDP programming, http://www.forum.nokia.com/html_reader/main/1,32611,2106,00.html?page_nbr=2

[32] Nokia Corporation (2002-12-19). J2ME MIDP Market & Technology Update, http://www.forum.nokia.com/seap/Java_Market_Update_Frederick_Westring2.pdf

[33] Nokia Corporation (2002-12-16). Phone Features – 3410, http://www.nokia.com/nokia/0,5184,539,00.html

[34] Northstream AB (2002-12-12). 3G strategies for operators, http://www.lucent.com/livelink/090094038000632f_Newsletter.pdf

[35] Northstream AB (2002-12-15). Mobile application download: key considerations for mobile operators http://www.northstream.se/download/applicationdownload.pdf

[36] Pollie, R. (2002-10-17). Write Once, Run Anywhere(TM)--Is It For Real? , http://java.sun.com/features/1997/aug/wora.html

[37] QUALCOMM Incorporated (2002-12-16). BREW at a Glance, http://www.qualcomm.com/brew/about/ataglance.html

[38] QUALCOMM Incorporated (2002-12-16). Developing for BREW, http://www.qualcomm.com/brew/developer/developing/developing.html

[39] QUALCOMM Incorporated (2002-12-16). The Road to Profit is Paved with Data Revenue, http://www.qualcomm.com/brew/about/whitepaper04.html

[40] Sony Ericsson Mobile Communications (2002-11-29), Intervju med Hanz Häger, Java Product Manager.

[41] Sun Microsystems AB (2002-12-28), Intervju med Lars Lundberg, Business Development Manager, Wireless Center of Excellence.

[42] Sun Microsystems Inc (2002-12-16). Big in Japan, http://java.sun.com/features/2001/03/docomo.html

[43] Sun Microsystems Inc (2002-12-15). Introduction to the Wireless Toolkit, http://java.sun.com/j2me/docs/alt-html/WTK104_UG/ch_intr.html#wp15039

[44] Sun Microsystems Inc (2002-10-24). Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices http://java.sun.com/products/cldc/wp/KVMwp.pdf

[45] Sun Microsystems Inc (2002-10-17). Java 2 Micro Edition (J2ME) Specifications. http://wireless.java.sun.com/midp/chapters/j2megiguere/chap4.pdf

[46]Sun Microsystems Inc (2002-12-15). Over The Air User Initiated Provisioning Recommended Practice for the Mobile Information Device Profile, http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf

[47] Sun Microsystems Inc (2002-10-23). PersonalJava(TM) Application Environment. http://java.sun.com/products/personaljava/spec-1-1-2/index.html

[48] Sun Microsystems Inc (2002-12-15). J2ME[oldtm] Optional Packages,
http://wireless.java.sun.com/midp/articles/optional/

[49] Sun Microsystems Inc. (2002-10-17). The Java(tm) Language: An Overview, http://java.sun.com/docs/overviews/java/java-overview-1.html

[50] Sun Microsystems Inc (2002-10-29). The JavaTM Product Versioning Specification
http://java.sun.com/products/jdk/1.2/docs/guide/versioning/spec/VersioningSpecification.html

[51] Sun Microsystems Inc (2002-12-15). User's Guide
Wireless Toolkit Version 1.0.4 Java™ 2 Platform, Micro Edition,
http://java.sun.com/j2me/docs/pdf/WTK104_UserGuide.pdf

[52] Sun Microsystems Inc (2002-10-23). Wireless Java(TM) Security
http://wireless.java.sun.com/midp/articles/security/

[53] Sun Microsystems Inc (2002-12-15). Wireless Application Programming MIDP Programming and Packaging Basics,
http://wireless.java.sun.com/midp/articles/getstart/

[54] Sun Microsystems Inc (2002-10-18). VM Spec Introduction,
http://java.sun.com/docs/books/vmspec/html/Introduction.doc.html

[55] Synergenix Interactive AB (2002-12-16). Mophun,
http://mophun.com/main_distributors.php?distrib_id=how

[56] Tele2 AB (2002-11-29), Intervju med Peter Storåkers, Produktchef för mobiltelefoni.

[57] Y Feng, J Zhu (2001). Wireless Java Programming with Java 2 Micro Edition.201 West 103rd Street, Indianapolis, IN, Sams Publishing

[58] Zelos Group LLC (2002-12-19). Zelos Group - Expertise - Roadmap Reports,
http://www.zelosgroup.com/expertise/wirelessjava.htm

Appendix A

Klasshierarki för Mobile Information Device Profile

- class java.lang.Object
  - class javax.microedition.lcdui.AlertType
  - class java.lang.Boolean
  - class java.lang.Byte
  - class java.util.Calendar
  - class java.lang.Character
  - class java.lang.Class
  - class javax.microedition.lcdui.Command
  - class javax.microedition.io.Connector
  - class java.util.Date
  - class javax.microedition.lcdui.Display
  - class javax.microedition.lcdui.Displayable
    - class javax.microedition.lcdui.Canvas
    - class javax.microedition.lcdui.Screen
      - class javax.microedition.lcdui.Alert
      - class javax.microedition.lcdui.Form
      - class javax.microedition.lcdui.List (implements javax.microedition.lcdui.Choice)
      - class javax.microedition.lcdui.TextBox
  - class javax.microedition.lcdui.Font
  - class javax.microedition.lcdui.Graphics
  - class java.util.Hashtable
  - class javax.microedition.lcdui.Image
  - class java.io.InputStream
    - class java.io.ByteArrayInputStream
    - class java.io.DataInputStream (implements java.io.DataInput)
  - class java.lang.Integer
  - class javax.microedition.lcdui.Item
    - class javax.microedition.lcdui.ChoiceGroup (implements javax.microedition.lcdui.Choice)
    - class javax.microedition.lcdui.DateField
    - class javax.microedition.lcdui.Gauge
    - class javax.microedition.lcdui.ImageItem
    - class javax.microedition.lcdui.StringItem
    - class javax.microedition.lcdui.TextField
  - class java.lang.Long
  - class java.lang.Math
  - class javax.microedition.midlet.MIDlet
  - class java.io.OutputStream
    - class java.io.ByteArrayOutputStream
    - class java.io.DataOutputStream (implements java.io.DataOutput)
    - class java.io.PrintStream
  - class java.util.Random
  - class java.io.Reader
  - class java.io.InputStreamReader
  - class javax.microedition.rms.RecordStore
  - class java.lang.Runtime

- class java.lang.Short
- class java.lang.String
- class java.lang.StringBuffer
- class java.lang.System
- class java.lang.Thread (implements java.lang.Runnable)
- class java.lang.Throwable
  - class java.lang.Error
    - class java.lang.VirtualMachineError
      - class java.lang.OutOfMemoryError
  - class java.lang.Exception
    - class java.lang.ClassNotFoundException
    - class java.lang.IllegalAccessException
    - class java.lang.InstantiationException
    - class java.lang.InterruptedException
    - class java.io.IOException
      - class javax.microedition.io.ConnectionNotFoundException
      - class java.io.EOFException
      - class java.io.InterruptedIOException
      - class java.io.UnsupportedEncodingException
      - class java.io.UTFDataFormatException
    - class javax.microedition.midlet.MIDletStateChangeException
    - class javax.microedition.rms.RecordStoreException
      - class javax.microedition.rms.InvalidRecordIDException
      - class javax.microedition.rms.RecordStoreFullException
      - class javax.microedition.rms.RecordStoreNotFoundException
      - class javax.microedition.rms.RecordStoreNotOpenException
    - class java.lang.RuntimeException
      - class java.lang.ArithmeticException
      - class java.lang.ArrayStoreException
      - class java.lang.ClassCastException
      - class java.util.EmptyStackException
      - class java.lang.IllegalArgumentException
        - class java.lang.IllegalThreadStateException
        - class java.lang.NumberFormatException
      - class java.lang.IllegalMonitorStateException
      - class java.lang.IllegalStateException
      - class java.lang.IndexOutOfBoundsException
        - class java.lang.ArrayIndexOutOfBoundsException
        - class java.lang.StringIndexOutOfBoundsException
      - class java.lang.NegativeArraySizeException
      - class java.util.NoSuchElementException
      - class java.lang.NullPointerException
      - class java.lang.SecurityException
- class javax.microedition.lcdui.Ticker
- class java.util.Timer
- class java.util.TimerTask (implements java.lang.Runnable)
- class java.util.TimeZone
- class java.util.Vector
  - class java.util.Stack
- class java.io.Writer
  - class java.io.OutputStreamWriter

Interfacehierarki för Mobile Information Device Profile

- interface javax.microedition.lcdui.Choice
- interface javax.microedition.lcdui.CommandListener
- interface javax.microedition.io.Connection
  - interface javax.microedition.io.DatagramConnection
  - interface javax.microedition.io.InputConnection
    - interface javax.microedition.io.StreamConnection(also extends javax.microedition.io.OutputConnection)
      - interface javax.microedition.io.ContentConnection
        - interface javax.microedition.io.HttpConnection
  - interface javax.microedition.io.OutputConnection
    - interface javax.microedition.io.StreamConnection(also extends javax.microedition.io.InputConnection)
      - interface javax.microedition.io.ContentConnection
        - interface javax.microedition.io.HttpConnection
  - interface javax.microedition.io.StreamConnectionNotifier
- interface java.io.DataInput
  - interface javax.microedition.io.Datagram(also extends java.io.DataOutput)
- interface java.io.DataOutput
  - interface javax.microedition.io.Datagram(also extends java.io.DataInput)
- interface java.util.Enumeration
- interface javax.microedition.lcdui.ItemStateListener
- interface javax.microedition.rms.RecordComparator
- interface javax.microedition.rms.RecordEnumeration
- interface javax.microedition.rms.RecordFilter
- interface javax.microedition.rms.RecordListener
- interface java.lang.Runnable

Appendix B
Frågor till Sun

Hur många jobbar med J2ME i er organisation? Typer av arbetsuppgifter?

Vilka är kunderna, vilken målgrupp riktar ni er till?

Varför utvecklar Sun java? (What's in it for you?)

Hur kommer spridningen av applikationerna att gå till? Vad kommer man att betala för? Hur betalar man?

Vilka typer av applikationer utvecklas nu, och hur ser framtiden ut?

Vilka kommer att leverera/utveckla nyttoapplikationer för J2ME?

Vilka är de största hoten mot J2ME? Vilka tekniker konkurrerar?

Hur ser samarbetet ut mellan Sun, telefontillverkare, telefonoperatörer och applikationsleverantörer? Hur vill ni att samarbetet skall gå till? Vad vill de andra parterna?

Vad är er roll i expertgruppen för MIDP? Vad har ni tryckt på eller tillfört? Vilken är arbetsgången i expertgruppen för MIDP?

Vilka förväntningar har ni på MIDP-NG och vad ser ni för möjligheter?

Upplever ni att det finns få utvecklare som satsar på J2ME?

Kommer J2ME att bli en kassako, och i så fall när? Och för vem?

Finns det en försiktighet i branschen, väntar ni på 3G, push-teknik, bättre hårdvara och så vidare?

Vilka tar initiativen? Vad är den drivande kraften i utvecklingen av ny teknologi som denna? Telefontillverkarna, operatörerna, utvecklarna eller Sun?

Vilka problem finns i produktutveckling när tekniken utvecklas snabbt? (MIDP1.0 -> 2.0) Är det viktigt för telefontillverkarna att klara den senaste specifikationen när nya modeller lanseras?

Kommer J2ME att bestå eller är det en övergående hype?

Appendix C
Frågor till telefontillverkare

Vad gjorde att ni började med J2ME?

Hur många jobbar med J2ME i er organisation? Typer av arbetsuppgifter?

Hur kommer det sig att uppslutningen inom industrin är så stor bakom J2ME? Vad är anledningen till att tredjepartstillverkare nu ”släppts in i” telefonerna?

Kommer ni lägga in mjukvara (MIDlets) som gör telefonen mer attraktiv för konsumenen?

Vad är er roll i expertgruppen för MIDP? Vad har ni tryckt på eller tillfört? Vilken är arbetsgången i expertgruppen för MIDP? What’s in it for you?

Vilka förväntningar har ni på MIDP-NG och vad ser ni för möjligheter?

Vilka är de största hoten mot J2ME? Vilka tekniker konkurrerar?

Vilka har anammat de första Javastödda telefonerna?

Vilka tror du kommer vara de största leverantörerna av J2ME spel?

Vilka kommer att leverera/utveckla nyttoapplikationer för J2ME?

Hur stor är efterfrågan, beräknad efterfrågan i framtiden?

Hur kommer spridningen av applikationerna att gå till? Vad kommer man att betala för?

Vilka typer av applikationer utvecklas nu, och hur ser framtiden ut?

Upplever ni att det finns få utvecklare som satsar på J2ME?

Kommer J2ME att bli en kassako, och i så fall när? Och för vem?

Hur kommer samarbetet att se ut mellan operatörer, telefontillverkare och applikationsleverantörer? Hur vill ni att samarbetet skall gå till? Är det genomförbart? Vad vill de andra parterna? Vad tjänar Sun på det hela?

Har ni samarbete med operatörer angående till exempel wap-gateways (filtyper, filstorlek)?

Vilka problem finns i produktutveckling när tekniken utvecklas snabbt? (MIDP1.0 -> 2.0) Är det viktigt att klara den senaste specifikationen när nya modeller lanseras?

Har ni tillägg till MIDP1.0 API:n som bara stödjs av era produkter? Eller vill ni bara ha produkter som följer standarden?

Kommer J2ME att bestå eller är det en övergående hype?

Appendix D
Frågor till utvecklare

Hur många jobbar med J2ME i er organisation? Typer av arbetsuppgifter?

Hur länge har ni utvecklat program i J2ME? Vad gjorde att ni började?

Vilka är kunderna, vilken målgrupp riktar ni er till?

Hur stor del av er verksamheten är att utveckla program för trådlösa enheter?

Vilka har anammat de första produkterna?

Vilka tror ni kommer vara de största leverantörerna av J2ME spel?

Vilka kommer att leverera/utveckla nyttoapplikationer för J2ME?

Hur stor är efterfrågan, beräknad efterfrågan i framtiden?

Hur kommer spridningen av applikationerna att gå till? Vad kommer man att betala för? Hur betalar man?

Vilka typer av applikationer utvecklas nu, och hur ser framtiden ut?

Kommer J2ME att bli en kassako, och i så fall när?

Vilka är de största hoten mot J2ME? Vilka tekniker konkurrerar?

Hur kommer samarbetet att se ut mellan operatörer, telefontillverkare och applikationsleverantörer? Hur vill ni att samarbetet skall gå till? Vad vill de andra parterna?

Har ni samarbete med operatörer angående till exempel wap-gateways (filtyper, filstorlek)?

Är det lätt att utveckla i J2ME? Är det lätt att komma igång? Relatera gärna till annan mjukvaruutveckling.

Vilka utvecklingsmiljöer använder ni er av? Till exempel Forte (ONE Studio), J2ME Wireless toolkit eller liknande. Vad är positivt respektive negativt med utvecklingsmiljön?

Vad skall göras på serversidan och vad skall göras på klienterna, enligt er uppfattning?

I vilken tycker form tycker ni att data skall skickas mellan telefon och server?

Utvecklar ni specifikt för olika telefonmodeller? Och i så fall, av vilken anledning?

Vilken påverkan har den begränsade processorkraften och den låga minneskapciteten för utvecklingen av MIDlets?

Hur sker testningen?

Vilka är skillnaderna i resultat och prestanda mellan tester i emulatorer och i telefonerna?

Går det att förutse flaskhalsar och i så fall, hur undviks de?

Vilka begränsningar i den färdiga applikationen beror på MIDP1.0 specifikationen?

Vilka förväntningar har du på MIDP-NG och vad ser du för möjligheter?

De applikationer ni gör, görs dessa parallellt i andra språk för olika plattformar och i så fall vilka?

Kommer J2ME att bestå eller är det en övergående hype?

Appendix E
Frågor till operatörer
Hur många jobbar med J2ME i er organisation? Typ av arbetsuppgifter?

Hur länge har ni arbetat med J2ME? Hur började det?

Hur kommer spridningen av applikationerna att gå till? Vad kommer man att betala för och hur? Kan man reklamera?

Vilka typer av applikationer utvecklas nu, och hur ser framtiden ut?

Upplever ni att det finns få utvecklare som satsar på J2ME?
Har dom för få kvalitativa tjänster?

Kommer J2ME att bli en kassako för er del, och i så fall när?

Vilka är kunderna, vilken målgrupp riktar ni er till? Ses J2ME som en del i att få kunder att teckna abonnemang, istället för att välja en annan operatör med ”fräckare” portal.

Efterfrågar kunderna javatjänster eller tillgång till dem, genom att till exempel be er släppa igenom .jar och .jad filer genom er wap-gateway?

Finns det en försiktighet i branschen (niten med wap), väntar ni på 3G?
Vad krävs för att ni skall våga satsa?
Är ni rädda att supportkostnaden kommer att skjuta i höjden?

Kommer java-applikationer ta stora delar av sms-marknaden (förtjänsten) i till exempel en chatfunktion (gprs data)?

Kommer ni att leverera tjänster/applikationer och i så fall av vilken typ? Prioriteras javatekniken högt inom operatörsföretagen? What’s in it for you?

Hur kommer samarbetet att se ut mellan operatörer, telefontillverkare och applikationsleverantörer? Hur vill ni att samarbetet skall gå till? Är det genomförbart? Vad vill de andra parterna?

Uppvaktas ni av applikation- och plattformstillverkare? Vad är det dom vill sälja?

Vilka tar initiativen? Vad är den drivande kraften i utvecklingen av ny teknologi som denna? Telefontillverkarna, operatörerna, utvecklarna eller användarna?

Vilka är de största hoten mot J2ME? Vilka tekniker konkurrerar?

Kommer J2ME att bestå eller är det en övergående hype?