TECHNISCHE UNIVERSITÄT CHEMNITZ
Fakultät für Informatik
Professur Rechnernetze und verteilte Systeme

Diplomarbeit

Synchronisation von Terminplanern mittels XML

von

Mirko Mrowczynski

Studiengang: Informatik

Vertiefungsgebiet: Rechnernetze

Betreuender Hochschullehrer: Prof. Dr.-Ing. habil. Uwe Hübner

Tag der Ausgabe: 01.03.2001

Tage der Abgabe: 30.10.2001

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Thema - Synchronisation von Terminplanern mittels XML

Terminplaner, Adressbuch-Anwendungen usw. werden derzeit auf einer Vielzahl unterschiedlicher Plattformen genutzt. Daher sind Abgleichverfahren sowohl für mehrere Rechner zu einer Person (Desktop ... PDA) als auch für Team-Koordinierungen nötig. Im Rahmen dieser Arbeit sollen die existierenden Standards und Lösungen analysiert werden. Aufbauend auf die gefundenen Defizite und Richtungen der Standardisierung (SyncML ...) ist ein eigener Prototyp zu realisieren (Plattformen, Werkzeuge und Implementierungssprachen sind mit dem Betreuer abzustimmen).

Kapitel 2

Bedarfsanalyse

2.1 Terminplaner Gestern und Heute

In der Terminplanung sollen Termine so koordiniert werden, das alle Termine wahrgenommen werden können. Es darf also keine zeitlichen Überschneidungen geben. Als Termin soll die Notwendigkeit gesehen werden, innerhalb einer Zeitspanne deren Dauer und Anfangspunkt feststeht, eine bestimmte Aufgabe zu lösen. Die Fähigkeit, Termine wahrnehmen zu können, steht in engem Zusammenhang mit Mobilität. Je schneller ich mich von einem Ort zum anderen bewegen kann, desto mehr Termine kann ich wahrnehmen.

Bei vielen Terminen müssen keine physischen Aktivitäten ausgeführt werden. Diese Termine beschränken sich auf Kommunikation. Je nach Anforderungen für diese Kommunikation und dem Grad der technischen Ausstattung der Teilnehmer ist demnach die persönliche Anwesenheit nicht immer nötig. Das kann den Aufwand für die Lösung der Aufgabe zeitlich verschieben.

Beispiele hierfür sind das Einreichen von Unterlagen bis zu einem bestimmten Zeitpunkt und der Telefontermin als Alternative zum persönlichen Erscheinen. Im ersten Fall kann die Zeit für das Lösen der Aufgabe frei zwischen dem Bekanntwerden der Aufgabe und dem Termin Ihrer Fälligkeit gewählt werden, und es wird je nach Art der Zustellung die Zeit für das persönliche Erscheinen gespart. Im zweiten Fall findet der Termin zwar zum festgelegten Zeitpunkt statt, der Ort kann aber relativ frei gewählt werden.

Hohe Mobilität und Kommunikationsmöglichkeiten verringern den Aufwand für Termine und sorgen dafür, das wir zunehmend mehr Termine pro Zeiteinheit haben können. Es ist ausserdem möglich, kurzfristiger zu planen.

Die Möglichkeit zu haben etwas tun zu können zwingt zwar nicht dazu es auch zu tun, doch mit der Möglichkeit geht meist auch die gedankliche Durchdringung einher und das sorgt dafür das sich bestimmte Sachen nach und nach durchsetzen. Die Modeerscheinung der 80er Jahre bei den Terminplanern war der Filofax. Ein Filofax ist Terminkalender, Notiz- und Adressbuch und ToDo-List in einem. Wer ihn hatte, notierte sich darin Alles, was sich in eine der eben beschriebenen Kategorien einordnen lies. Je nach Umfang konnte ein Filofax ein beachtliches Gewicht erreichen. Das und die Tatsache, das die einzigen

“Backupmöglichkeiten” die von Xerox-Kopierer waren, sorgte dafür das sich in der 90ern Elektronische Geräte etablieren konnten. Papierorganizer werden heute zielgerichteter eingesetzt und sind weniger umfangreich.

2.2 Besitz und Einsatz eines Terminplaners

Im privaten Bereich beginnt der Terminplaner bei einem Zettel auf dem Termine für einen folgenden, meist eng abgesteckten Zeitraum, notiert werden. Sind die Termine vorbei verschwindet auch der “Terminplaner”. Häufig verwendet und auf Dauer angelegt sind dagegen Geburtstagskalender. Feierlichkeiten und Gratulationen zählen auch zu den Terminen. Von grossem Vorteil ist hierbei die Wiederverwendbarkeit bei Eintragung in einen Jahreskalender. Eher wiederkehrende Ereignisse (14tägig, monatlich) lassen sich oft nur schlecht auf Papier automatisieren und müssen von Hand nachgetragen werden. Terminplaner für die laufenden einmaligen Termine setzt jeder nach seinen persönlichen Vorlieben ein, wobei elektronische Planer den Vorteil haben, das sie an Termine erinnern ohne das der Besitzer nachsieht. Solange das Hausaufgabenheft und der Stundenplan in der Schule aus Papier sind, wird diese Form der Terminplanung sicher eine hohe Akzeptanz beibehalten. Das macht späteres Umlernen auf elektronische Möglichkeiten der Terminplanung erforderlich und birgt eventuell den Nachteil, das der Anwender von seiner bisherigen Terminplanung voreingenommen ist. Ergebnis ist dann die bevorzugte Verwendung von Applikationen die eins zu eins von Papier in die Elektronik umgesetzt sind. Ein Wecker dient übrigens auch der Terminplanung. Damit lässt sich zwar nur ein Termin festlegen, der dann auch noch innerhalb der nächsten 12 Stunden ist, aber bei vielen Mobiltelefonen und PDA´s lässt sich nur über den Terminplaner eine Weckfunktion realisieren. Geweckt werden ist also ein spezieller Termin.

Terminplaner Privat	Terminplaner Firma
Papier und elektronisch	meist Papier, selten elektronisch
nur Besitzer ändert Daten; Einsicht durch Andere unerwünscht	Änderung durch jeden Nutzer oder eine autorisierte Person
je nach System gute Synchronisation möglich	Synchronisaton schwer koordinierbar; wird dem Anwender überlassen
technisch dem Kenntnisstand des Einzelnen angepasst	muss (meist intuitiv) durch alle Anwender bedienbar sein
Funktionalität wichtiger als Sicherheit	Schutz vor Verlust durch Redundanz

Tabelle 2.1: Eigenschaften von Planern in verschiedenen Umfeldern

Im Umfeld von Firmen sieht man häufig Terminkalender auf Papier. Das sind meist grosse Jahreskalender für die Urlaubsplanung und Wochen- oder Monatskalender für den Dienstplan der Mitarbeiter. Diese posterähnlichen Wandkalender bieten jedem Mitarbeiter einen guten Zugriff. Sie können jeder Zeit gelesen werden und für das Beschreiben sind meist

Regeln festgelegt. Das kann so aussehen, das jeder seinen Vorschlag als Entwurf einträgt und dann gemeinsam unter Einfluss einer Kontrollinstanz (dem Chef) der endgültige Plan entsteht. Eine Übertragung der Daten die den Einzelnen interessieren in dessen privaten Terminplan, erfolgt durch abschreiben. Änderungen im Plan können nur für den Zeitraum vorgenommen werden, ab dem die Beteiligten die Änderung zur Kenntnis nehmen. Elektronische Möglichkeiten der Terminplanung werden in Firmen (noch) eher selten eingesetzt. Damit verbundene Vorteile, wie das schnelle Übernehmen der Daten in den eigenen Plan und Aktualisierung über ein Netzwerk in kurzen Abständen, sind offensichtlich. Nachteile wie hohe Anschaffungskosten für die Ausrüstung und die fehlende Erfahrung, wie sich ein solches System in der Praxis verhält, wiegen jedoch im Moment viele Vorteile auf.

2.3 Wer verwendet / benötigt einen Terminplaner

Seit ich mich mit Terminplanern beschäftige, halte ich auch nach Statistiken darüber Ausschau, wer einen Terminplaner verwendet. Ich finde es wäre interessant zu wissen, welche Zielgruppe welche Art von Terminplaner bevorzugt. Das kann zum einen davon abhängen, welcher Art die Termine sind oder wie sie vereinbahrt werden (schneller Zugriff, Suchmöglichkeiten, Wiederholungen, akkustische Benachrichtigung, ..). Andererseits ist die Anzahl der Termine und daraus resultierend die mit dem Terminplaner zugebrachte Zeit sicher ein wichtiges Kriterium für die Anschaffung.

Eine qualitative und quantitative Analyse über die Verwendung könnte sowohl dem Suchenden als auch einem Verkäufer hilfreiche Richtlinien liefern. Bisher erfolgt die Beratung meist bezüglich des Speichers, manchmal wird noch auf die Displaygrösse eingegangen. Da der Verkäufer normalerweise keine Erfahrung mit dem speziellen Gerät, hat bleiben Informationen zu Stromverbrauch und Synchronisationsmöglichkeiten aussen vor. Ganz zu Schweigen vom Gesamteindruck den man bekommt, wenn das Gerät an seiner Kapazitätsgrenze betrieben wird.

Wer auf der Suche nach einem Papier-Terminplaner ist, sollte sich folgende Fragen stellen :

- Wieviele Termine pro Tag / Woche / Monat ? (entscheidend für die Aufteilung; bei weniger als 1 Termin pro Zeiteinheit, kann die nächst grössere Übersicht gewählt werden.)

- Telefon- und Adresseinträge möglich ? (Umheften von Seiten die länger als der Terminplaner gültig sind)

- Beschränkung auf Geschäfts oder Privatbereich ? (z.B. nur Geschäftsbereich, Terminplaner kann im Büro bleiben und entsprechend grösser und detailierter ausfallen)

- Zeitliche Einschränkungen aus Platzgründen möglich ? (weniger Platz nachtsüber, am Wochenende)

- Qualität bezüglich äusserer Einflüsse (Wasser-, Schmutz-, Lichtbeständigkeit). Das kann unter Umständen der Grund sein sich für einen Planer aus Papier zu entscheiden !

Für den Interessenten an einem elektronischen Terminplaner ist folgendes wichtig:

- Mobilität gefordert ? (Programme für den PC sind komfortabel, aber ohne Notebook bleibt der Terminplaner zu Hause oder im Büro.)
- Wird eine Wochen- oder gar Monatsübersicht benötigt ? (Viele Planer haben nur eine Tagesübersicht)
- Archivierung alter Termine (Drucken, PC-Übertragung)
- Backupmöglichkeiten zur Datensicherung (Speicherkarte, PC-Übertragung)
- Synchronisation mit anderen Terminplanern nötig ? (auch Adressaustausch)
- Terminplaner als eigenständiges Gerät oder in ein anderes integriert ? (Der Trend geht zu Kombigeräten. Da dort der Terminplaner oft nur ein Anhängsel ist, leidet darunter seine Qualität.)
- Einspielen neuer Software nötig ? (Firmenspezifische Software, Updates, Bugfixes)
- Einsatzort klären (Beleuchtung, Tastatur / Display, Temperatur!)
- Stromlose Speicherung der Daten ? (Batteriewechsel, länger nicht benutzt)

Ich habe bei meiner Recherche keine Analyse im Sinne von “diese Gruppe benutzt diesen Planer” oder “sollte diesen Planer verwenden” gefunden. Die meisten Untersuchungen stützen sich auf Verkaufszahlen und treffen damit keine Aussage über die Zufriedenheit der Kunden.

Kapitel 3

Übersicht über aktuelle Terminplaner

3.1 Desktopsysteme

3.1.1 Gnome-PIM

Gnome-PIM steht für Gnome Personal Information Manager. Bevor ich mich mit ihm befasse, möchte ich kurz auf die Entstehungsgeschichte von Gnome eingehen. Gnome [9] bedeutet GNU Network Object Model Environment und ist Teil des GNU Projektes. Im GNU Projekt [10] wird seit 1984 an frei verfügbaren UNIX-Quellen gearbeitet. Das Gnome Projekt wurde gegründet, um dem Nutzer eine einfach zu benutzende Arbeitsoberfläche und die dazu passenden Applikationen bereitzustellen. Bei der Entwicklung wurde darauf geachtet, das Gnome auf verschiedenen Hardwareplattformen und unterschiedlichen UNIX Systemen läuft. Da Gnome eine freie Software mit offen liegenden Quellen ist und beliebig abgeändert und verteilt werden darf, beteiligen sich viele Leute an der Entwicklung. Dadurch entsteht innerhalb kurzer Zeit ein hochwertiges Produkt und auch Fehler werden schnell gefunden und behoben, da viel getestet wird. Weil Entwickler und Nutzer verschiedenster Nationalitäten beteiligt sind, erfolgte von Anfang an auch eine Internationalisierung. Gnome Applikationen gibt es deshalb in vielen Sprachen, was besonders für eine schnelle Einarbeitung und Akzeptanz wichtig ist.

“gnomecal” ist der Terminplaner des Personal Information Manager. Auf den ersten Blick macht dieser Planer einen eher spartanischen Eindruck. Beim Arbeiten merkt man jedoch, das nur häufig gebrauchte Funktionen direkt verfügbar sind und sich alles Weitere über gut strukturierte Menüs erreichen lässt. Sehr positiv fallen die permanent verfügbaren Schaltflächen Tages-, Wochen-, Monats- und Jahresübersicht auf. Mit diesen Übersichten kann auf einen Blick festgestellt werden, ob und welche Termine an einem Tag anliegen. Auch die Alarmfunktion geht über ein einfaches Klingeln hinaus. So kann zum eingestellten Termin ein vom Nutzer angegebenes Programm gestartet, oder eine E-Mail versandt werden. Es ist auch möglich, einem Termin eine bestimmte Kategorie zuzuweisen. Die hierfür wählbaren Möglichkeiten öffentlich, privat und vertraulich sind meiner Meinung nach jedoch unglücklich gewählt, oder vielleicht auch nur übersetzt. Sie sind leider nicht editierbar und in der mir vorliegenden Version 1.0.55 lässt sich auch kein Filter einschalten, um z.B.

nur Termine einer Kategorie anzuzeigen. Viel Wert wurde auf die Festlegung wiederkehrender Termine gelegt. Mit nur wenigen Mausklicks kann man Wiederholungen wie “jeden Dienstag und Donnerstag aller zwei Wochen” oder “jeden dritten Mittwoch im Monat” festlegen.

Über den Aufbau und die Bestandteile von Gnome wurde in der zweiten Ausgabe des Linux-Magazins von 1999 sehr ausführlich berichtet [11]. Zu diesem Zeitpunkt war der Gnome Desktop in Version 1.0 fast fertiggestellt. Mit seiner Veröffentlichung wurde mit der Entwicklung einer Office-Suite begonnen. Die Applikationen, welche in dieser Office-Suite enthaltenen sind, wurden zum Teil zeitgleich mit dem Gnome-Desktop und den zugehörigen Bibliotheken entwickelt. Als Kalender- , Mail- und Adresstool steht in der Office-Suite die Applikation “Evolution” zur Verfügung. Es handelt sich hierbei noch um eine Beta Version, die jedoch bezüglich ihrer Kalender und Adresskomponenten auf die weit vorangeschrittenen Applikationen “gnomecal” und “gnomecard” zurückzuführen ist.

Auch der “korganizer” im KDE ist eigentlich ein “gnomecal”. Nur das KDE Anwendungen auf der QT Bibliothek basieren während Gnome eine eigene CORBA Implementierung zu Grunde liegt. Da beide Terminplaner das vCalendar Format unterstützen, ist auch der Austausch von Termindaten kein Problem.

3.1.2 plan

Das Programm “plan” ist ein Terminplaner für Linux. Es sind verschiedene Kalenderansichten möglich, bei denen zwischen grossen Übersichten und vielen Details gewählt werden kann. Für eine bestimmte Ansicht können Einstellungen per Koniguratio nsdatei geändert werden. Das ist z.B. für die Anpassung an regionale Feiertage sinnvoll.

Alarme, als Hinweis auf das Eintreten eines Termins, können verschiedene Aktionen ausführen. Möglich ist das Senden einer Mail, Starten eines Shell-Skripts oder Anzeigen eines Fensters auf dem Bildschirm. Ein sehr schöne Funktion ist die Ankündigung von Terminen. Das ist besonders bei Terminen, für die eine Vorbereitung nötig ist, hilfreich.

Mit den bis jetzt genannten Features ist plan ein Terminplaner unter vielen. Eine Funktion, die das Programm heraushebt, ist die Möglichkeit, Termine von Gruppen verwalten zu können. Der Austausch von Gruppendaten erfolgt entweder über ein Verzeichnis, das allen zugänglich ist, oder über einen Server, der die Koordination übernimmt. Der Server informiert die Terminplaner der Gruppenmitglieder über Änderungen und beugt dadurch Inkonsistenzen zwischen Schreibzugriffen verschiedener Nutzer vor. Das Nutzen der Termindaten auf dem Apple Newton und dem Palm PDA ist duch Zusatzprogramme möglich.

Eine gute Beschreibung zu plan ist in einer älteren Ausgabe des Linux Magazin [31] zu finden. Auch wenn es dort nicht um die aktuelle Version von plan geht, sind die hervorzuhebenden Funktionen beschrieben. Die letzte veröffentlichte Version ist 1.8.4 von Juli 2000.

3.2 Terminplaner auf mobilen Geräten

3.2.1 Casio Diary

An den “Diary´s” von CASIO kann man schön die Entwicklung mobiler Geräte mit Terminplanerfunktion sehen. Die ersten Geräte kamen Anfang bis Mitte der neunziger Jahre auf. Sie hatten verglichen mit heutigen Geräten eine geringe Speicherkapazität und eine kleine Anzeigefläche. Das Modell C-300 war z.B. mit 4kB Arbeitsspeicher und einem 4x16 Zeichen Display ausgestattet. An Software war vieles integriert, was auf aktuellen Geräten auch zu finden ist. Durch die deutlich spürbare Resourcenknappheit fallen diese Programme teilweise jedoch sehr spartanisch aus. So fehlt im Memopad eine Suchfunktion und im Terminkalender sieht man jeweils nur einen Termin. Das damit verbundene Umblättern ist eher contraproduktiv.

Hard- und Software waren zu diesem Zeitpunkt so eng verschmolzen, das selbst Produkte eines Herstellers in der Bedienung nur Ähnlichkeiten, aber kaum Gemeinsamkeiten aufwiesen. Schon durch unterschiedliche Displaygrössen mussten die reichlich verwendeten Icons ständig angepasst werden. Auch die ständig mitwachsende Speichergrösse wirkte sich auf die Kompatibilität nachteilig aus. Grössere Speicher erforderten andere Datenstrukturen. Während bei den ersten Geräten ein Adressbucheintrag ein einziges Textfeld ist, wurde später stärker strukturiert.

Aktuelle Vertreter dieser Produktgruppe haben bis zu 2 MB Arbeitsspeicher und auch das Display ist im Laufe der Zeit gewachsen. Und zwar auf 8 Zeilen a 40 Zeichen. Auch in diesen Geräten ist die Software vorgegeben und kann nicht durch den Benutzer gewählt werden. Angesichts der geringen Stückzahlen eines Gerätetypes ist es auch fraglich, ob sich ein Markt für die Erstellung und den Austausch von herstellerunabhängiger Software bilden würde.

Die einzige Möglichkeit den Speicher zu vergrössern oder Programme hinzuzufügen ist bei einigen Modellen durch plug-in Cards gegeben. Diese Karten haben eine Kapazität von 64kByte bis 256kByte und sind bezüglich Form und Anschlüssen eine Entwicklung von CASIO, lassen sich also auch nur in diesen Geräten einsetzen. Auf diesen Karten wird nicht viel, aber sehr nützliche Software angeboten, die durch ihre Grösse nicht serienmässig auf Diary´s installiert ist. Angeboten werden z.B. mehrsprachige Wörterbücher, Thesaurus, wissenschaftliche Taschenrechner und eine Tabellenkalkulation.

Auch wenn der Trend ein anderer ist, hat die nichtaustauschbare Unterbringung von Applikationen Vorteile. Programme die in einem ROM-Speicher liegen, benötigen bei abgeschaltetem Gerät keinen Strom. Die Batterien werden nur dafür verwendet, die durch den Nutzer eingegebenen Daten zu erhalten. Die Betriebsdauer kann daher bei einer Nutzung von nur wenigen Minuten täglich mehr als ein Jahr betragen. Im unbenutzten Zustand werden Informationen im Speicher über Jahre gehalten. Für einen Terminplaner ist das nicht so entscheidend, aber bei Adressbuchfunktionen liegt der Vorteil sicher auf der Hand.

Hoch anzurechnen ist CASIO, das alle Geräte mit einer seriellen Schnittstelle ausgestattet sind. Das ermöglicht eine unkomplizierte Übertragung von einem Gerät auf ein

Anderes. Die Geräte müssen dabei nicht vom selben Typ sein, aber zur selben Familie gehören (gleiche Buchstaben am Anfang der Typbezeichnung).

Wenn eine Übertragung zwischen Geräten erfolgen soll, die nicht direkt miteinander kommunizieren können, so ist das mittels PC möglich. Die PC-Software filtert dabei die empfangenen Daten und ermöglicht dadurch das Übertragen an andere Gerätefamilien.

In der Praxis kann es jedoch schwierig werden, die zur Übertragung nötige Hard- und Software zu installieren. Die seriellen Schnittstellen der Geräte haben einen 2.5 oder 3.5 mm Klinkenstecker und arbeiten mit unterschiedlichen Spannungen. Diese Spannungen unterscheiden sich noch dazu von der einer RS232 Schnittstelle am PC. Die benötigten Kabel enthalten deshalb aktive Bauelemente um die Spannungen anzupassen. Bei einigen Kabeln wird nun die zur Versorgung dieser Bauelemente benötigte Energie über unbenutzte Datenleitungen des PC geliefert. Bei anderen Kabeln ist eine separate Stromversorgung über Battarie oder Netzteil vorgesehen. Im Programm muss also eingestellt werden, welches Kabel und welches Gerät verwendet wird. Viele Programme unterstützen aber leider nur einen Teil der existierenden Hardware. Sie sind deshalb oft nicht in der Lage mit dem vorhandenen Gerät zu kommunizieren oder die ausgelesenen Daten in das Format des zweiten Diary zu übertragen.

Wer an frei verfügbarer Software zu diesem Thema interessiert ist findet unter http://www.casiocorner.rpd.nl/programs_pc.htm eine grosse Auswahl. Ein Programm das neben CASIO Geräten auch eine Reihe von Sharp Organizern unterstützt ist XLink/Win [2]. Es handelt sich um eine Windows Applikation, die herstellerübergreifend Daten zwischen diesen Geräten austauschen kann.

Ein Diary prüft beim Einspielen von Datensätzen nicht, ob das ankommende Datum bereits im Speicher vorhanden ist. Diese Redundanzkontrolle obliegt der Software auf dem PC. Wenn mehrere Datensätze mit gleichem Inhalt ankommen, werden auch mehrere Datensätze mit gleichem Inhalt gespeichert.

Die in den vorangegangenen Abschnitten am Beispiel der CASIO Diary´s gezeigte Entwicklung ist die Entwicklung eines ganzen Marktes. Andere Hersteller haben ähnliche Produkte auf den Markt gebracht und sicher auch ähnliche Probleme zu lösen gehabt. Ich habe mich bei der Beschreibung für CASIO-Produkte entschieden, weil hier die Vielzahl der Geräte besonders gross ist und sie über einen langen Zeitraum entstanden sind. Ein Entwicklungsprozess ist dadurch besonders gut erkennbar.

Mit der Pocket-PC Produktlinie Casiopeia löst sich CASIO vom Diary Design als aufklappbares Gerät mit Tastatur und Display. Der Casiopeia E115G als erster Vertreter dieser Gruppe ist mit einem 320x240 Pixel grossen berührungsempfindlichen Farbdisplay ausgestattet und auf eine Tastatur wird ganz verzichtet. Die Beschreibung solcher Geräte mit dem Schwerpunkt Terminplanung möchte ich jedoch nicht am Casiopeia durchführen. Dafür eignet sich die Palm-Reihe von 3COM meiner Meinung nach besser.

3.2.2 Palm Pilot

Der erste Palm, der “Pilot 1000”, wurde von Jeff Hawkins entwickelt und 1996 von US-Robotics auf den Markt gebracht [3]. Ich empfehle dem Leser den Namen des Pilot Entwicklers in eine Suchmaschine einzugeben und einige der erscheinenden Berichte und Interviews zu lesen. Jeff Hawkins hat mit den Pilot, Palm und Handspring PDA´s einen ganzen Markt revolutioniert oder erst geschaffen und ist es, finde ich, wert, das man einige Zeilen über ihn liest. US-Robotics wurde kurz nach Einführung des Pilot von 3COM gekauft und der Name Pilot wurde wegen rechtlicher Probleme aufgegeben. Neue Geräte haben als Namen “Palm” gefolgt von einer Versionsnummer und werden von 3Com produziert.

Dem Palm-Projekt wurde anfangs wenig Erfolg prophezeit, da Apple eine Schlappe mit seinem Newton [4]erlitten hat. Dazu nur folgendes : Apple Chef Mike Spindler ruft seinen Vorgänger John Sculley an. "John, wir müssten uns mal zusammensetzen. Wie wärs morgen um 15 Uhr?" "Einen Moment, ich muß erst in meinem Newton Message-Pad schauen." (Eine Minute Pause.) "Nein, Mike. Um drei habe ich keine Zeit. Da habe ich einen Termin mit Kfd@hk Mbrinsdt in der Wqbrw\c-Bar. “[5]

Dieser Witz bringt ein Problem der damaligen PDA´s (Personal Digital Assistant) auf den Punkt. Sie waren zu kompliziert zu bedienen und teilweise unzuverlässig. Der Buchstabensalat im obigen Zitat spielt auf die Handschrifterkennung des Newton an. Hinzu kam Eigenschaften wie Grösse, Gewicht und Batterielebensdauer. Jeff Hawkins hatte sich also einiges vorgenommen, um trotz des schlechten Images von PDA´s ein erfolgreiches Produkt auf den Markt zu bringen. Sein Konzept basierte darauf sich eng am Nutzer orientierend auf Wesentliches zu beschränken. Der erste Pilot hatte nur die Applikationen Adressbuch, Kalender, Memo Pad, Merkzettel und Rechner. Das ist weniger als andere Produkte zu bieten hatten. Der Entwicklung ging jedoch eine Marktanalyse voran und da hat sich gezeigt, das es die Applikationen sind, mit denen der Nutzer die meiste Zeit zubringt. Die geringen Ausmasse und die lange Betriebszeit mit einem Satz Batterien haben ein übriges getan, um den Absatz zu fördern.

Die Energie bezieht ein Palm normalerweise aus zwei LR03 Monozellen. Das reicht für etwa 40 Betriebsstunden oder 2 Monate Standby. Dank eines eingebauten Kondensators übersteht das Gerät auch einen Batteriewechsel ohne Datenverlust [6]. Eine eigene Energiequelle für den Speicher gibt es jedoch nicht. Wenn die Batterien nicht rechtzeitig getauscht werden, kommt es zum Datenverlust. Deshalb wurde von Anfang an Wert auf Sicherungsmöglichkeiten gelegt. Alle Geräte wurden daher mit einem Cradle, einer Halterung für den PC, mit seriellem oder USB Anschluss, ausgestattet. Die mitgelieferte Software unterstützt das Abgleichen von Daten zwischen PC und Palm, Eingabe und Änderung von Datensätzen und das Aufspielen neuer Software.

Der Palm bietet die Möglichkeit, zusätzlich zu den im ROM-Speicher untergebrachten Applikationen, weitere in den Arbeitsspeicher des Gerätes zu installieren. Der Nutzer erhält dadurch die Möglichkeit, Betriebssystemupdates durchzuführen und aus dem reichen Softwareangebot zu schöpfen, das Dank der Freigabe eines Entwicklerkits entstanden ist. Für Palm-Geräte gibt es inzwischen Tausende von Applikationen, die von der Tabellenkalkulation über Tauchplaner bis hin zum Game-Boy Emulator reichen. Interessant dabei ist, das es nicht gelungen ist eine der ursprünglichen Applikationen durch "third-party" Produkte zu ersetzten. Einzige Ausnahme bildet für einige Nutzer der Taschenrechner, der in seiner ursprünglichen Form keine wissenschaftlichen Funktionen enthält. Beim Taschenrechner fallen jedoch auch keine Datenmengen an, die intern oder auf dem PC gespeichert werden sollten. Bei Adress- oder Termindaten dagegen spielt die Synchronisation mit dem PC, oder besser noch, mit den dort verwendeten Adressdatenbanken und Terminplanern eine Grosse Rolle. Andere Datenformate im Palm würden Ergänzungen in der Übertragungssoftware und im verwendeten Terminplaner nach sich ziehen.

Die Technik zum Synchronisieren zwischen PC und PDA heisst bei Palm HotSync. Zum Synchronisieren wird der PDA in das Cradle gesteckt und dort, per Knopfdruck auf die HotSync-Taste, der Datenabgleich eingeleitet. Das Cradle vom ersten Pilot 1000 passt übrigens auch für alle Modelle bis zum Palm III und auch der Palm V kann es mit einem rein mechanischen Adapter verwenden (Einen Palm IV gab es nie, weil das im asiatischen Raum als Unglückszahl gilt). Auf dem PC muss der mitgelieferte HotSync-Manager installiert sein. Wenn der Hot-Sync-Manager am PC bei Systemstart automatisch gestartet wird muss zum Synchronisieren nicht einmal der Monitor eingeschaltet werden.

Der Hot-Sync-Manager liest die Datenbanken des Gerätes und übergibt sie je nach Typ einem zuständigen Conduit. Conduits sind Programmteile, die Daten für eine Anwendung auf dem PC aufbereiten. Es gibt sie für verbreitete Windows Programme, wie MS-Outlook und auch unter Linux im KPilot und gnome-pilot [7]. Diese Conduits sorgen natürlich auch dafür, das am PC erstellte oder aktualisierte Daten zur Übertragung an den Palm vorbereitet werden. Das ist besonders bei Mails und News interessant, weil der Informationsfluss hier fast ausschlieslich in dieser Richtung erfolgt. Wird für eine Datenbank kein spezielles Conduit gefunden, wird eines verwendet, das die Datenbank unverändert auf dem PC sichert, um vor Verlust zu schützen.

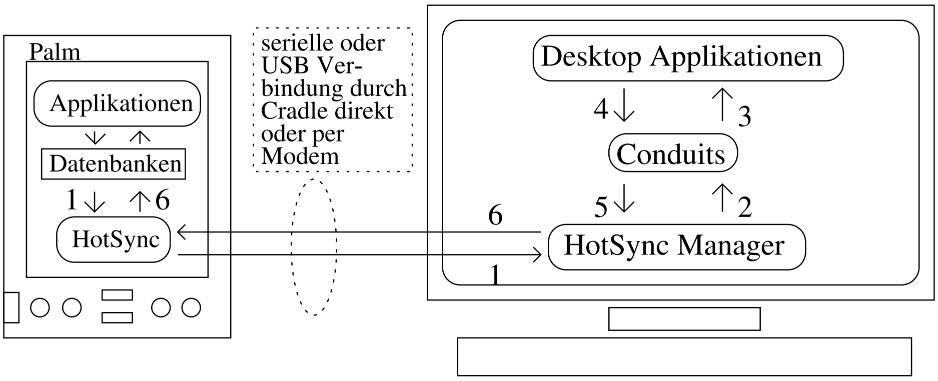


Abbildung 3.1: Zeitliche Abfolge zweier Synchronisationen während derer die Palmdatenbanken am PC bearbeitet werden

Palm Geräte bieten also eine recht komfortable Möglichkeit Termin- und Adressdaten zwischen einem mobilen Gerät und einer Desktopapplikation konsistent zu halten. Es ist dabei auch kein Problem, ein Palm Gerät und mehrere PC´s, z.B. zu Hause und im Büro, zu verwenden. Wenn an allen Orten mit dem Palm synchronisiert wird, stimmen auch die Termine auf den PC´s überein. Auch mehrere mobile Geräte am PC zu synchronisieren ist mit einem HotSync Manager möglich. Zu beachten ist dabei jedoch, das bei Windows Systemen jeder Nutzer Zugriff auf die Daten anderer Nutzer hat.

Zum Abgleich vieler PDA´s an einem Cradle (oder wenigen Cradles) gibt es von Palm den HotSync Server [16]. Der Server ist besonders für Firmen interessant, die ihre Mitarbeiter mit PalmOS PDA´s ausstatten. Dadurch kann allen Nutzern einheitliche Funktionalität zur Verfügung gestellt werden und gleichzeitig wird der administrative Aufwand minimiert. Auch die Bereitstellung von Conduits für firmenspezifische Applikationen ist möglich. Durch die Auswertung statistischer Daten, wie z.B. wann und wo wird was aktualisiert, kann der Einsatz der Geräte sogar noch an die Anforderungen in der Firma angepasst werden. Mit einem zentralen Server können aber auch private Daten der Anwender gespeichert werden. Da Sicherheit beim Palm immer als “Safety” und nicht als “Security” zu verstehen ist hat der Anwender auch kaum eine Chance sich dagegen zu schützen. Das Synchronisieren von Windows CE Geräten mit dem HotSync Server ist geplant. In wie weit damit Daten zwischen CE und Palm Geräten ausgetauscht werden können, bleibt abzuwarten. Ein PalmOS Gerät mit Geräten die ein anderes Betriebssystem verwenden abzugleichen, ist jedoch nicht ohne weiteres möglich. Da die Interoperabilität jedes dieser Produkte betrifft, habe ich diesem Sachverhalt Abschnitt 3.3 gewidmet.

HotSync wird nicht nur bei Palm, sondern bei allen Geräten mit dem PalmOS eingesetzt. Die grössten Lizenznehmer für PalmOS sind Handspring und IBM, die ihre VISOR und WorkPad PDA´s damit betreiben. Weil nicht jeder, der sich einen Palm zulegt, bereits ein Office Paket sein Eigen nennt und es nicht für alle Pakete Conduits gibt, steht dem Palm Besitzer der “Palm Desktop” als PIM auf dem PC zur Verfügung. Diese Software kann in ihrer aktuellen Version kostenlos geladen werden und arbeitet mit den vom HotSync Manager synchronisierten Daten. Es ist möglich, die Datenbanken für Terminkalender, Adressbuch, ToDo Liste und Notitzbuch zu editieren. Ausserdem können Palm Applikationen ausgewählt werden, die bei der nächsten Synchronisation auf dem mobilen Gerät installiert werden sollen. Möglichkeiten, Softwareupdates und Datenbankeinträge für alle auf diesem PC verwalteten Mobilgeräte vorzunehmen, sind aber nicht vorgesehen. Es wird immer auf der Datenbasis eines Gerätes gearbeitet. Das Gerät wird dabei am Namen des Nutzers manuell identifiziert.

3.2.3 Nokia Mobiltelefone

Das man mit Mobiltelefonen mehr machen kann als anzurufen, ist für uns selbstverständlich. Mobiltelefone mit 7-Segment-LED-Anzeige kennen wir meist nur aus amerikanischen Filmen. Selbst alte Modelle aus der Zeit als das Autotelefon gerade tragbar wurde und in die Jackentasche passte, hatten mindestens eine Telefonbuchfunktion. Seit dem hat sich einiges getan. Einige Mobiltelefone haben inzwischen zweistellige Menüpunkte. Es war also nur eine Frage der Zeit, oder besser der Resourcen, bis zum Terminplaner im Mobiltelefon.

Nokia Mobiltelefone haben seit 1998 ab Modell 6110 einen Terminplaner. Grösste "Handicaps" bei der Benutzung stellen die kleine Anzeige und die schwer zu bedienende Tastatur dar. Nokia hat deshalb viele Funktionen zur Bedienung in Menüs untergebracht und den Terminplaner stark strukturiert. Bis auf den Namen des Termines und die Uhrzeit wann er stattfindet, lässt sich der Terminplaner mit vier Tasten bedienen.

Nach dem Aufruf des Kalendermenüs sieht man das Kalenderblatt des aktuellen Tages. Der Terminplaner unterscheidet die Kategorien Erinnerung, Anruf, Sitzung und Geburtstag. Sollte am angezeigten Tag für eine dieser Kategorien ein Termin vorliegen, so wird das durch ein Symbol angezeigt.Wenn der Nutzer für eine Kategorie täglich Termine hat, verliert das Symbol jedoch seine Aussagekraft, da es nur angibt, das mindestens ein Termin vorliegt. Ähnlich ist die Problematik beim Planen neuer Termine. Das der Nutzer immer nur Informationen über einen Tag oder gar nur einen Termin an diesem Tag erhält, erleichtert die Planung sicher nicht. Es fehlt, wohl vor allem durch die geringe Displaygrösse, eine Möglichkeit Übersichten zur Auslastung anzuzeigen. Um zu ermitteln an welchem Tag der kommenden Woche ein geeigneter Termin für eine bestimmte Sache ist, muss der Nutzer auf den kommenden Montag vorblättern, sich die Notitzen zeigen lassen, auf Dienstag vorblättern usw. Diese Methode zu aufwändig um z.B. während eines Telefongespräches, das dann auch noch mit Freisprecheinrichtung oder Headset geführt werden müsste, einen günstigen Termin zu finden. Ein Terminplaner wie er in den Nokia 6100 Mobiltelefon integriert ist, lässt sich demnach nur bis zu einer bestimmten Anzahl von Terminen sinnvoll einsetzen. Der Nutzer sollte in einem solchen Fall wissen wo zeitlich Engpässe auftreten und vom Terminplaner überwiegend die Erinnerungsfunktion nutzen. Dem Kalender im Telefon steht eine Speicherkapazität von weniger als 2kByte für maximal 100 Termine zur Verfügung [8]. Aus Gründen der Übersichtlichkeit ist es jedoch nicht empfehlenswert, diesen Wert auszuschöpfen. Der Terminplaner im Telefon sollte daher als jederzeit zu Verfügung stehende Ergänzung zu anderen Systemen gesehen werden.

Nokia bietet zur Kommunikation mit dem PC unter Win95/98/NT die Nokia Data Suite an. Sie beinhaltet das Verbindungskabel zwischen serieller Schnittstelle am PC und dem Mobiltelefon.

Bei der folgenden Generation von Nokia Mobiltelefonen wurde die Menüführung zugunsten der häufiger benötigten Funktionen geändert. Der Punkt Drucken über Infrarot, der nur mit der passenden Empfängerhardware möglich ist und daher kaum benötigt wird, ist im Modell 6210 nicht mehr vorhanden. Dank des grösseren Displays konnte hier jedoch eine Wochenübersicht im Kalender eingeführt werden.

3.3 Systemübergreifende Synchronisierung

In diesem Kapitel habe ich bisher gezeigt, welche Hard- und Softwarelösungen zur Terminplanung existieren und welche Kommunikationsmöglichkeiten es zwischen diesen gibt. Da jeweils ein spezielles Produkt behandelt wurde, lag die Betonung immer auf der Kommunikation zu ergänzender Hard- oder Software. Es wurden Lösungen behandelt wie man zwischen PDA´s und Office Paketen Daten austauscht, weil diese Funktionalität meist zum Umfang der mitgelieferten Software des Anbieters gehört. Wie aber unterschiedliche PDA´s direkt miteinander kommunizieren wurde nicht gezeigt. Das liegt daran das die technischen Realisierungen sich so unterscheiden, das eine direkte Kommunikation überhaupt nicht möglich ist, weil die Übertragungsschnittstellen keinen Anschluss zulassen und die implementierten Protokolle sich nicht verstehen. Ein Datenaustausch zwischen zwei in der Funktion ähnlichen, aber in der Realisierung verschiedenen Geräten, erfolgt daher immer über eine dritte Instanz, den PC.

Auch mit Hilfe eines resourcenstarken Computers bleibt die Datenübertragung zu einem grossen Teil Handarbeit. Die mit einem mobilen Gerät vertriebene Software kennt nur dieses Gerät und ermöglicht nur die Datenanbindung an ihr bekannte Bürosoftware. Aus dieser Bürosoftware können die Informationen dann vom Programm des zweiten Geräten abgeholt und auf dieses Gerät übertragen werden. Selbst bei einem einmaligen Datenaustausch, z.B. bei der Anschaffung eines neuen Gerätes, ist diese Prozedur anstrengend. Für einen regelmässigen Datenaustausch ist dieses Verfahren nicht praktikabel.

Bei der Beantwortung der Frage, warum die Hersteller elektronischer Terminplaner ihren Kunden den Datenaustausch mit Fremdgeräten nicht einfacher machen, bin ich auf folgende Antworten gestossen: Weil Firmen ökonomischen Zwängen unterliegen ist, jeder erst einmal bestrebt sein eigenes Produkt zu vermarkten und als Standard zu etablieren. Er versucht das für das gesamte Produkt inclusive Datenformaten und Schnittstellenspezifikationen. Das es noch andere Produkte für dieselbe Zielgruppe gibt wird schlicht ignoriert. Intern müssen sich die Firmen natürlich mit ihrer Konkurrenz auseinandersetzen. Nach aussen erfährt der Kunde jedoch nichts davon.

Wenn die im Gerät gesammelten Informationen für den Nutzer wichtig sind, eine Übertragung an ein ähnliches Gerät eines anderen Herstellers aber schwierig, dann erfolgt auch eine Bindung an Produkte dieses Herstellers. Deshalb sind die Anbieter bestrebt eine möglichst breite Basis für den Einsatz ihrer Hard- und Software zu schaffen. Das zeigt sich bei der Anbindung an Office Pakete und an der Menge der verfügbaren Zusatzhardware. In die Tiefe zu gehen und möglichst viele verschiedene Geräte am Markt zu etablieren, ist dagegen nicht von Interesse.

Die Erstellung von Software, die den Datenaustausch zwischen Geräten unterschiedlicher Hersteller übernimmt, ist also eine Nische die von den Geräteherstellern selbst nicht ausgefüllt wird. In dieser Nische hat sich jedoch eine Reihe von Firmen etabliert. Mit dem Programm Intellisync der Firma Pumatech [17] lassen sich z.B. PDA´s mit PalmOS und Windows CE synchronisieren. Intellisync setzt dabei die Synchronisationssoftware HotSync und ActiveSync voraus, verwendet diese für den Datenaustausch, verwaltet aber anschliessend die übertragenen Datenobjekte selbst. Etwas anders funktioniert die iMobile Suite von der Firma Synchrologic [18]. Hier erfolgt der Datenaustausch mit einem eigenen auf dem PDA zu installierenden Clienten. Der Aufwand der damit betrieben wird ist deutlich höher als bei den meisten anderen Lösungen. Durch einen eigenen Clienten konnte auch die Funktionalität erhöht werden. Es sind z.B. automatische Backup´s möglich.

Aus technischer Sicht sind die beiden vorgestellten Produkte Extreme. Im ersten Fall wird auch auf der PC Seite die originale Software für den Datenaustausch verwendet. Im zweiten Fall wird auf dem mobilen Gerät ein eigener Client installiert. Die meisten anderen angebotenen Lösungen bewegen sich insofern dazwischen, das sie dem mobilen Gerät keine weiteren Fähigkeiten hinzufügen, auf dem PC dagegen Übertragung und Verwaltung der Daten mit der eigenen Software lösen.

Alle vorgestellten Lösungen sind aber darauf ausgelegt, einem Benutzer den Austausch seiner Daten zu ermöglichen. Bei der Synchronisation erfolgt ein merge, ein Mischen der Daten von Quelle und Ziel. Sollte das Ziel keine Daten enhalten, entsteht eine Kopie der Quelldaten. Wenn mit den vorgestellten Lösungen Daten zu einer anderen Prerson übertragen werden sollen, funktioniert das bei Adressdatenbanken noch recht praktikabel, schlimmstenfalls erhält der Empfänger mehr Daten als erforderlich. Das Mischen von Kalenderdaten verschiedener Personen ist dagegen sinnlos. Hier wäre ein Vergleichen der Daten wichtig, um einen gemeinsamen Termin zu finden. Wenn zu Semesterbeginn jeder Student am Anfang einer interfakultären Veranstaltung seinen Terminkalender bereitstellen würde, dann wäre es möglich die Terminfindung zu automatisieren. Das würde Zeit in der Veranstaltung sparen und, je nach Lösungen, mehr Leuten den Besuch der Veranstaltung ermöglichen. Ein solches Projektmanagement gibt es leider nicht bzw. ist der Aufwand bezüglich der Zeit für die Datenaufbereitung und Kosten grösser als der Nutzen.

Die folgenden Relationen zwischen Anwendern und Geräten sind bezüglich des Datenaustausches möglich:

Ein Anwender, ein Produkt

- Der Anwender erstellt seine Daten auf diesem Gerät oder mit dieser Software.
- Die Verwendung der Daten in einem anderen Gerät oder einer anderen Software ist nicht vorgesehen.
- Ausnahme ist die Verwendung in einer anderen Instanz des gleichen Gerätes oder der gleichen Software für die keine Transformation der Daten nötig ist.
- Datenübertragung wird (wenn überhaupt) ausserhalb des Produktes initiiert. (Datei kopieren; Speichermedium wechseln; abschreiben)

Ein Anwender, unterschiedliche Produkte

- Der Anwender arbeitet mit unterschiedlichen Produkten aber gleichen Daten.
- Die Arbeit geschieht zeitlich nacheinander.
- Um Daten in einem anderen Produkt nutzen zu können ist eine Transformation erforderlich.
- Datenübertragung wird mit Hilfe des Produktes durchgeführt (HotSync, ActiveSync und Conduits)

Mehrere Anwender, ein Produkt(typ)

- Die Anwender arbeiten auf unterschiedlichen Daten.
- Daten werden nur selektiv und nicht komplett übertragen.
- Es ist keine Transformation der Daten nötig. Richtlinien, wie bei Überschneidungen von Objektbezeichnern oder Zeiten vorzugehen ist, sind erforderlich.
- Die Datenübertragung wird mit Hilfe des Produktes durchgeführt. Weil es sich um nur einen Gerätetyp handelt, kann das auch ohne dritte Instanz, direkt erfolgen (z.B. Infrarot).

Mehrere Anwender, verschiedene Produkte

- Die Anwender arbeiten mit Geräten verschiedenen Typs und auf der Basis unterschiedlicher Daten.
- Nur ein Teil der Daten wird übertragen. Bei der Übertragung ist eine Transformation vom Quell- ins Zieldatenformat nötig.
- Die Datenübertragung erfolgt eher selten mit Hilfe der Produkte. Es gibt keine allgemeine Lösung für den Datenaustausch in diesem Szenario.

Kapitel 4

SyncML - ein XML basiertes Synchronisationsformat

Im vorherigen Kapitel habe ich gezeigt, das es eine Reihe praktikabler Lösungen zum Thema Terminplanung gibt. Auch Möglichkeiten Termin- und Adressdaten zwischen verschiedenen Systemen auszutauschen, bestehen. Der Datentransfer ist aber immer nur zwischen wenigen Geräten, Applikationen und Systemen möglich.

In den letzten Jahren hat sich die Extensible Markup Language (XML) in der Industrie und in Geschäftsbereichen als Standard für den Austausch und die Verteilung von Informationen etabliert. In meiner Arbeit geht es aber nicht um die Synchronisation von Daten im Allgemeinen, sondern vorrangig um Informationen für oder von Personal Information Managern. Das sind Adress- oder Termindaten die zwischen den Anwendungen eines Anwenders übertragen werden müssen oder für die Koordinierung eines Teams von Leuten nötig ist.

Um auf diesem Gebiet erstmals einen herstellerübergreifenden Standard zu entwickeln, schlossen sich führende Hersteller von Mobiltelefonen und Handheld Geräten zu einer Initiative zusammen. Inzwischen sind über 600 Firmen am SyncML Projekt beteiligt [12].

4.1 Was ist SyncML

SyncML ist ein Protokoll zur Datensynchronisation. Es kann von jedem Gerät, jeder Applikation auf jedem System verwendet werden. Die Synchronisation ist überall über jedes verfügbare Netzwerk möglich. Mit diesem Protokoll ist es möglich, alle persönlichen Informationen abzugleichen. Neben dem Terminkalender betrifft das vor allem E-Mail, Adressbücher und ToDo-Listen.

Die folgende Abbildung beschreibt die Funktionsweise von SyncML. SyncML ist als Client- Server Lösung geplant. Es ist jedoch auch möglich, zwei Clienten miteinander zu verbinden. Das ist zum Beispiel interessant um zur Einigung auf einen Termin zwei Terminkalender zu vergleichen, oder Visitenkarten auszutauschen. Auch Szenarien bei denen Server miteinander verbunden werden, sind denkbar. Das kann nötig sein, weil sich ein Client, je nach Standort und damit verbundener Netzinfrastruktur, an unterschiedliche Server wendet, die dann ihrerseits Daten austauschen. Aber auch das Bearbeiten von Nutzerdaten die auf unterschiedlichen Servern gespeichert sind, ist ein mögliches Einsatzgebiet. In der folgenden Abbildung sind die Funktionseinheiten hinter denen sich sowohl Client als auch Server verbergen können, als Unit A und B gekennzeichnet.

Wegen der knappen Resourcen bezüglich des Speicherplatz ist das mobile Gerät in der Regel Client. Der Client überlässt so viel Funktionalität wie möglich dem Server. Da zum Synchronisieren nur eine “Sync Engine” nötig ist wird darauf in der Regel beim Clienten verzichtet.

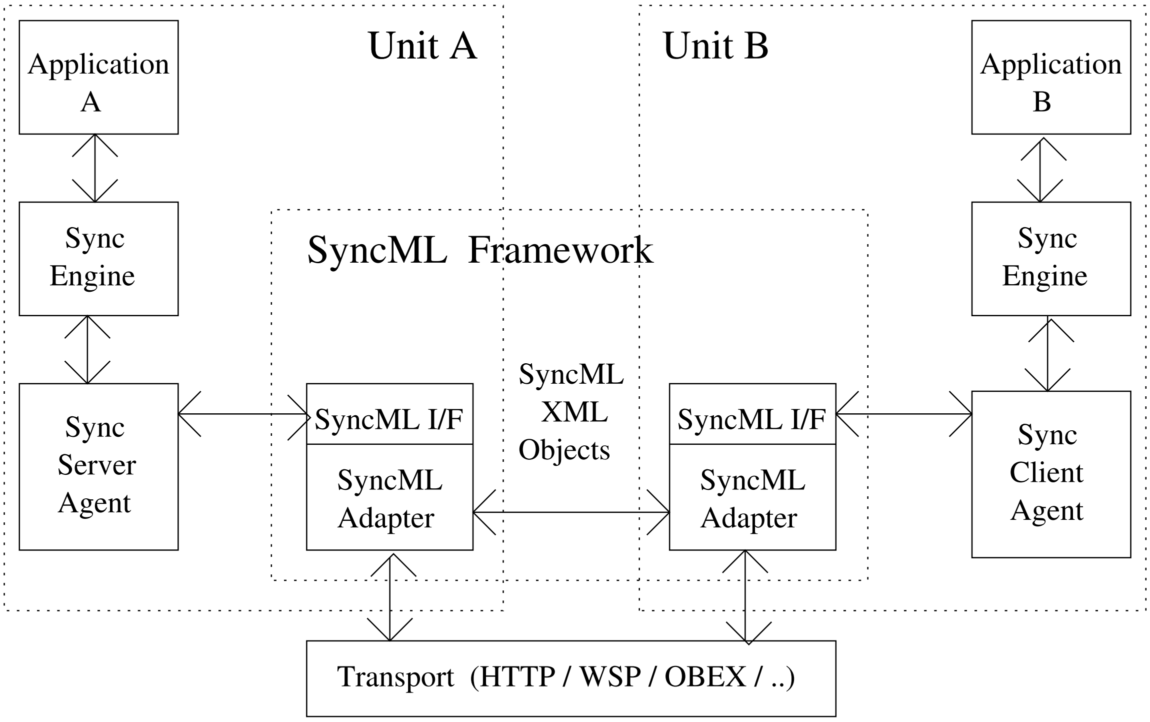


Abbildung 4.1: SyncML Strukturnetzwerk

Der direkte Austausch von SyncML Objekten zwischen SyncML Adaptern, wie in der Abbildung oben beschrieben, wird sicher nur selten angewendet werden. Dazu müssten Client und Server als Applikationen auf der selben Maschine laufen. Viel häufiger wird es vorkommen, das Client und Server auf unterschiedlichen Geräten laufen. Diese Geräte sind dann über ein Netzwerk verbunden. Die SyncML Objekte werden also durch den entsprechenden Netzwerkprotokollstack hinabgereicht, transportiert und auf der Empfängerseite zum SyncML Adapter geleitet. Die Daten werden aber entweder direkt von Applikation zu Applikation, oder über eine Netzwerk ausgetauscht. Eine Möglichkeit beide in der Abbildung gezeigten Übertragungswege gleichzeitig zu Nutzen, besteht nicht.

4.2 Mark-up Sprachbeschreibung

Im folgenden liste ich Elemente von SyncML auf, wie sie im Representation Protocol [13] beschrieben sind. Das soll einen Einblick in die Struktur der Sprache und die Mächtigkeit ihrer Funktionalität geben. SyncML Implementierungen beinhalten unter Umständen nicht den gesamten Umfang der Sprachelemente. Das gilt besonders für Client-Lösungen, die für eine Synchronisierung nur einen Teil der Funktionalität verwenden. Mit dieser Einschränkung lässt sich Speicherplatz im Gerät und Zeit für die Entwicklung sparen. Bezahlt wird das unter Umständen damit, das der Client nicht mit jedem SyncML Server synchronisieren kann.

4.2.1 Protokoll Befehlselemente

Add - Empfänger fügt der Datenbank des Absenders etwas hinzu

Alert - Möglichkeit um (Status-)Informationen an eine Applikation beim Empfänger zu schicken

Alert wird bei der Initialisierung der Synchronisation verwendet. Über die folgenden Alert Codes wird festgelegt wie synchronisiert werden soll.

TWO-WAY	Vom Clienten initiierte zwei Wege Synchronisation.
SLOW SYNC	Vom Clienten initiierte zwei Wege Synchronisation, bei der alle Daten übertragen werden.
ONE-WAY FROM CLIENT	Vom Clienten initiierte Einwegsynchronisation des Servers.
REFRESH FROM CLIENT	Vom Clienten initiierte Aktualisierung des “ONE-WAY FROM CLIENT”.
ONE WAY FROM SERVER	Vom Clienten initiierte Einwegsynchronisation des Clienten.
REFRESH FOM SERVER	Vom Clienten initiierte Aktualisierung des “ONE-WAY FROM SERVER”.
TWO-WAY BY SERVER	Vom Server initiierte zwei Wege Synchronisation.
ONE-WAY FROM CLIENT BY SERVER	Vom Server initiierte Einwegsynchronisation des Servers.
REFRESH FOM CLIENT BY SERVER	Vom Server initiierte Aktualisierung des “ONE-WAY FROM CLIENT”.
ONE-WAY FROM SERVER BY SERVER	Vom Server initiierte Einwegsynchronisation des Clienten.
REFRESH FROM SERVER BY SERVER	Vom Server initiierte Aktualisierung des “ONE-WAY FROM SERVER”.

Tabelle 4.1: Synchronisationstypen

Atomic - spezifiziert eine Reihe von SyncML Befehlen, die dann zusammen ausgeführt werden

Copy - kopiert Datenelemente innerhalb der Datenbank des Empfängers

Delete - löscht Daten

Exec - zum Ausführen eines Prozesses auf dem Gerät des Empfängers

Get - überträgt Daten vom Empfänger

Map - Befehl zum Hinzufügen oder Löschen von Objektbezeichnertabellen im Empfänger

MapItem - von Map verwendet, um Quell- Zielbeziehungen festzulegen

Put - überträgt Daten zum Empfänger

Replace - ersetzt Daten im Empfänger

Results - Liefert Ergebnisse zu einem Search oder Get Befehl. Im MsgRef Element wird der Bezeichner der Anfrage mit zurück geliefert.

Search - zum Suchen in der Datenbank des Empfänger

Sequence - ordnet die Ausführung einer Reihe von SyncML Befehlen

Sync - löst eine Operation zur Datensynchronisierung aus

4.2.2 Protokoll Management Element "Status"

Da "Status" das einzige Protokoll Management Element ist, fällt es entsprechend umfangreich aus. Status berichtet über den Stand der Ausführung eines SyncML Befehls. Über Befehle innerhalb des Befehls zu dem "Status" gehört wird keine Aussage getroffen. Hierauf beziehen sich eigene Status Elemente. Wenn in einer SyncML Anfrage mehrere Befehle enthalten waren, dann müssen die zugehörigen Status Antworten die gleiche Reihenfolge haben. Die vom "Status" Element gelieferten Status Codes sind dreistellig, wobei die erste Ziffer die Gruppe angibt, zu der der Code gehört.

1xy	informierend (Anfrage in Arbeit)
2xy	Anfrage erfolgreich ausgeführt
3xy	Umleitung (Ziel nicht erreichbar)
4xy	Fehler durch Absender
5xy	Fehler beim Empfänger

Tabelle 4.2: Arten von Status Codes

Das folgende SyncML Beispiel ist eine Antwort auf den Versuch, einer Datenbasis etwas hinzuzufügen. CmdID ist der eindeutige Bezeichner für diesen Befehl. MsgRef benennt die MsgID des Befehls, auf den sich Status bezieht. CmdRef benennt die CmdID des Befehls auf den sich Status bezieht. Mit Cmd wird der Name des Befehls angegeben auf den sich Status bezieht. Hier ist einer der Protokoll-Befehlselemente aus dem vorangegangenen Unterpunkt oder das Nachrichtenrahmenelement SyncHdr anzugeben. Data liefert die eigentliche Statusinformation. Nummer 403, ein Fehler durch den Absender, bedeutet das die Anfrage verstanden wurde, der Befehl hier jedoch verboten ist.

```
<SyncBody>
  <Status>
    <CmdID>1234</CmdID>
    <MsgRef>1</MsgRef>
    <CmdRef>5678</CmdRef>
    <Cmd>Add</Cmd>
    <Data>403</Data>
  </Status>
</SyncBody>
```

Abbildung 4.2: Beispiel mit “Status” zu einer “Add” Anfrage

4.2.3 Datenbeschreibungselemente

Die folgenden Elementtypen dienen als Rahmenelemente für die in einer SyncML Nachricht augetauschte Daten.

Data - spezifiziert einzelne SyncML Daten

Item - spezifiziert den Container (Rahmen) für ein Datenelement

Meta - spezifiziert Metainformationen über einen Eltern-Elementtyp (Oberklasse)

4.2.4 Nachrichtencontainerelemente

SyncML - spezifiziert den Container (Rahmen) für eine SyncML Nachricht und hat kein Elternelement

SyncHdr - spezifiziert den Container zum Überprüfen der SyncML Nachricht und für Routing Informationen

SyncBody - spezifiziert den Container für die eigentliche Nachricht

```
<SyncML xmlns=’SYNCML:SYNCML1.0’>
  <SyncHdr>
    ..
  </SyncHdr>
  <SyncBody>
    ..
  </SyncBody>
</SyncML>
```

Abbildung 4.3: Beispiel für die Schachtelung der Nachrichtencontainerelemente

4.2.5 Gemeinsam benutzte Elemente

Archive - zeigt an, das bei “Delete” spezifizierte Daten vor dem Löschen archiviert werden sollen

Chal - spezifiziert die Authentifikationsanforderung (Empfänger legt Typ und Format für die nächste Anfrage fest)

Cmd - benennt den SyncML Befehl auf den sich “Status” bezieht

CmdID - spezifiziert einen einheitlichen Befehlsbezeichner für diese Nachricht

CmdRef - benennt die “CmdID” auf die sich “Status” bezieht

Cred - spezifiziert den Berechtigungsnachweis für den Absender

Final - gibt an, das im aktuellen SyncML Paktet keine weiteren SyncML Nachrichten folgen

Lang - spezifiziert die bevorzugte Sprache für Befehle die Textnachrichten ausgeben

LocName - legt den Namen fest, der an Stelle der Ziel- oder Quelladresse gezeigt werden soll

LocURI - gibt die Ziel- oder Quelladresse an

MsgID - spezifiziert einen einheitlichen Nachrichtenbezeichner für diese Sitzung

MsgRef - benennt die “MsgID” auf die sich “Status” oder “Results” bezieht

NoResp - gibt an, das der Absender keinen “Status” in der Antwortnachricht übermittelt bekommen möchte

NoResults - Ergebnisse müssen an Empfänger weitergeleitet werden um als Eingabe für einen weiteren Befehl zu dienen

RespURI - spezifiziert die URI, die der Empfänger für Antworten auf diese Nachricht verwendt werden muss

SessionID - benennt einen Bezeichner für die SyncML Sitzung in der die Nachricht enthalten ist

SftDel - Datenelement wird beim Clienten gelöscht, aber bei der Synchronisierung weiter beachtet

Source - spezifiziert source routing oder mapping Informationen (URI , IMEI)

SourceRef - spezifiziert die Quelle auf die sich “Status” oder “Result” bezieht

Target - spezifiziert source routing oder mapping Informatinen (URI , IMEI)

TargetRef - spezifiziert das Ziel auf das sich “Status” oder “Result” bezieht

VerDTD - spezifiziert die Versionsnummer des SyncML Representationsprotokolles nach dem die Nachricht aufbereitet ist

VerProto - spezifiziert die Versionsnummer des SyncML Synchronisationsprotokolles nach dem die Nachricht aufbereitet ist

```
<SyncHdr>
  <VerDTD>1.0</VerDTD>
  <VerProto>SyncML/1.0</VerProto>
  <SessionID>1</SessionID>
  <MsgID>1</MsgID>
  <Target>
    <LocURI>http://pumuckl.csn.tu-chemnitz.de/servlet/sync/</LocURI>
  </Target>
  <Source><LocURI>IMEI: 493006106752323</LocURI></Source>
</SyncHdr>
```

Abbildung 4.4: SyncML Header mit Versions- und Sequencenummern, Ziel- und Quellangaben

4.3 SyncML Sync Protocol

So wie http, das “Hyper Text Transfer Protocol”, den Datenaustausch zur Übertragung von html, der “Hyper Text Markup Language” definiert ist, ist auch im SyncML Sync Protocol [14] beschrieben, wie die an Hand des SyncML Reference Protocol erzeugten Daten transferiert werden. Das Reference Protocol kümmert sich also um das “Was” während im Sync Protocol das “Wie” behandelt wird.

4.3.1 Grundlagen

Die Geräte müssen aufzeichnen, welche Veränderungen zwischen den Synchronisierungen an der Datenbank vorgenommen wurden. Das ist nötig, um zum Zeitpunkt der eigentlichen Synchronisierung nur die Änderungen übertragen zu müssen. Wenn ein Gerät mit mehreren Anderen synchronisiert werden muss, dann müssen die Unterschiede zum Datenbestand jedes der anderen Geräte festgehalten werden. Wie diese Aufzeichnung der geänderten Daten erfolgt, ist nicht spezifiziert.

Die Geräte vergleichen mit Sync Ankern, auf welchem Stand der Synchronisation das jeweils andere Gerät ist. Ein solcher Anker ist ein Zeitstempel. Beim Initialisieren werden der Anker der letzten Synchronisation und der aktuelle Anker übertragen. Stimmen die Anker überein, dann werden die Änderungen abgeglichen. Stimmen sie nicht, kann durch "Slow Sync" die komplette Datenbank übertragen werden.

Client und Server können für gleiche Datenelemente unterschiedliche Bezeichner verwenden. Es ist davon auszugehen, das für die lokalen Bezeichner des Clienten kürzere Einträge verwendet werden. Der Server setzt seine globalen Bezeichner, für Datenelemente, auf die lokalen Bezeichner des Clienten um. Diese Umsetzung wird als "ID Mapping" bezeichnet. Wenn der Server ein neues Datenelement zum Clienten überträgt, erhält er von diesem den lokalen Bezeichner für das übertragene Element, und kann seine Mapping-Tabelle aktualisieren.

Ist ein Datenelement auf der Server- und auf der Clientenseite verändert worden, entscheidet die Sync Engine wie weiter zu verfahren ist. Da der Server eine Sync Engine enthält entscheidet er normalerweise auch, was mit welchem der beiden Datenelemente passiert. Es ist jedoch möglich, das auch der Client Sync Engine Funktionalität enthält und sich an der Entscheidungsfindung beteiligt.

Um einen Server anzusprechen, kann vom Clienten eine URI basierte Adressierung verwendet werden. Dazu sollte der aufgerufene Dienst permanent verfügbar sein. Geräte die nur zeitweise angebunden sind, können sich selbst durch einen geeigneten Mechanismus identifizieren (IMEI des Mobiltelefons). Die Verwendung relativer Adressen für den Zugriff auf Datenbanken ist möglich, wenn der komplette Pfad auf den sich diese Adresse bezieht vorher genannt wurde.

Server:

```
..<Source>
  <LocURI>http://pumuckl.csn.tu-chemnitz.de/servlet/sync</LocURI>
</Source>..
```

Client:

```
..<Target>
  <LocURI>./calendar/user_mmr</LocURI>
</Target>..
```

entspricht absoluter Adresse:

```
..<Target>
  <LocURI>http://pumuckl.csn.tu-chemnitz.de
/servlet/sync/calendar/user_mmr</LocURI>
</Target>..
```

Der Client muss dem Server bei der ersten Synchronisation seine Geräteeigenschaften mitteilen. Auch bei Änderungen der Eigenschaften oder einer Anfrage durch den Server werden die Geräteeigenschaften mitgeteilt. Der Client sollte neben unveränderlichen Eigenschaften dynamische Informationen, wie verfügbare Speicherkapazität, mitteilen. Das kann zu Beginn jeder Synchronisation erfolgen.

Es ist möglich, eine Nachricht auf mehrere Packete aufzuteilen. Der Grund hierfür kann das Transportprotokoll oder der eingeschränkte Speicher eines beteiligten Gerätes sein. Im Unterschied zur Reasemblierung von Packeten in anderen Netzwerkprotokollen müssen bei SyncML nachfolgende Packete explizit vom Empfänger angefordert werden. Der Empfänger erhält damit die Möglichkeit, empfangene Daten während der Synchronisation zu verarbeiten, und dadurch Speicher zu sparen. Das letzte zu einer Nachricht gehörende Packet muss </Final> enthalten.

Es ist möglich, zusammen mit dem Initialisierungspacket Daten zu senden. Das reduziert die Anzahl der zu sendenden Packete, bringt aber auch Nachteile mit sich. So werden Daten übertragen, bevor die Sync Anker ausgetauscht sind. Wenn die Anker von Server und Client nicht übereinstimmen und ein Slow Sync eingeleitet wird, dann werden diese Daten ein zweites Mal gesendet. Ein grösserer Nachteil ist jedoch, das die Übertragung von Daten vor der Authentifizierung erfolgt. Der Client kann in dem Augenblick nicht sicher sein, das er mit dem richtigen Server kommuniziert.

Wenn der Server nicht innerhalb einer bestimmten Zeit auf die Anfrage eines Clienten reagieren kann, muss er den Clienten darüber informieren. Die Zeit in der diese Benachrichtigung zu erfolgen hat, hängt vom verwendeten Transportprotokoll und -medium ab. Der Client erhält die Information in einem Paket als Statusmeldung, die sich auf seine Anfrage bezieht. Der Client hat nach Empfang einer solchen Meldung die Möglichkeit, zu einem späteren Zeitpunkt nach den Ergebnissen zu fragen. Den Zeitpunkt muss er selbst wählen, vom Server erhält er hierzu keine Angaben. Die Anfrage nach Ergebnissen beim Server erfolgt über einen “Result Alert”. Der Client hat aber auch die Möglichkeit diese Synchronisation abzubrechen und zu einem späteren Zeitpunkt erneut zu starten. In diesem Fall werden auch die alten Sync Anker beibehalten.

4.3.2 Authentifikation

Authentifikation ist der Nachweis der Identität gegenüber dem Kommunikationspartner [15]. Dieser Identitätsnachweis kann in beide Richtungen erfolgen. Auch bei SyncML kann sich sowohl der Client gegenüber dem Server als auch der Server gegenüber dem Client authentifizieren. Beim Initiieren der Verbindung wird davon ausgegangen, das keine Authentifikation erforderlich ist. Sollte dies doch der Fall sein, dann teilt der Server das dem Clienten gegenüber mittels Statusmeldung mit. Möglich sind die Meldungen, das eine Authentifikation erforderlich oder eine erfolgte Authentifikation fehlgeschlagen ist. Um die Synchronisation zu initiieren, muss der Client das Paket daraufhin noch einmal mit gültiger Authentifikation schicken. Dazu wird das “Cred” Element aus dem Representation Protocol verwendet.

Der Berechtigungsnachweis erfolgt prinzipiell durch Nutzername und Password. Für die Übertragung sind zwei Varianten möglich. Im ersten Fall, der Authentifikation vom Typ “auth-basic” werden Nutzername und Password im Base64 Format übertragen. Bei der Base64 Codierung wird der 8-Bit Zeichensatz, einschliesslich nicht druckbarer Zeichen, auf einen 6-Bit Zeichensatz, der aus Zeichen des Alphabetes besteht, abgebildet. Das dient ausschliesslich der problemlosen Übertragung von Sonderzeichen und nicht der Erhöhung der Sicherheit. Base64 codierter Text ist wie Klartext zu behandeln. Im zweiten Fall, der Authentifikation vom Typ “auth-md5”, wird an Stelle von Nutzernamen und Passwort, der mit dem MD5 Algorithmus berechnete Hashwert übertragen. Die zur Berechnung des Wertes verwendete Hashfunktion ist nicht umkehrbar. Der Server hat keine Möglichkeit aus dem übertragenen Hashwert auf Nutzername und Password zu schliessen. Weil er Nutzername und Password kennt, kann er jedoch den Hashwert selbst berechnen und den berechneten mit dem vom Clienten übertragenen Wert vergleichen. Um zu verhindern das ein erlauschter Hashwert von einem Angreifer zu einem späteren Zeitpunkt zur Anmeldung missbraucht wird, sendet der Server einen Zufallswert an den Clienten. Server und Client hängen diesen Zufallswert, vor der Hashwertberechnung, an Nutzername und Passwort an. Damit wird bei jeder Authentifikation ein anderer Hashwert übertragen und eine sichere Authentifikation gewährleistet.

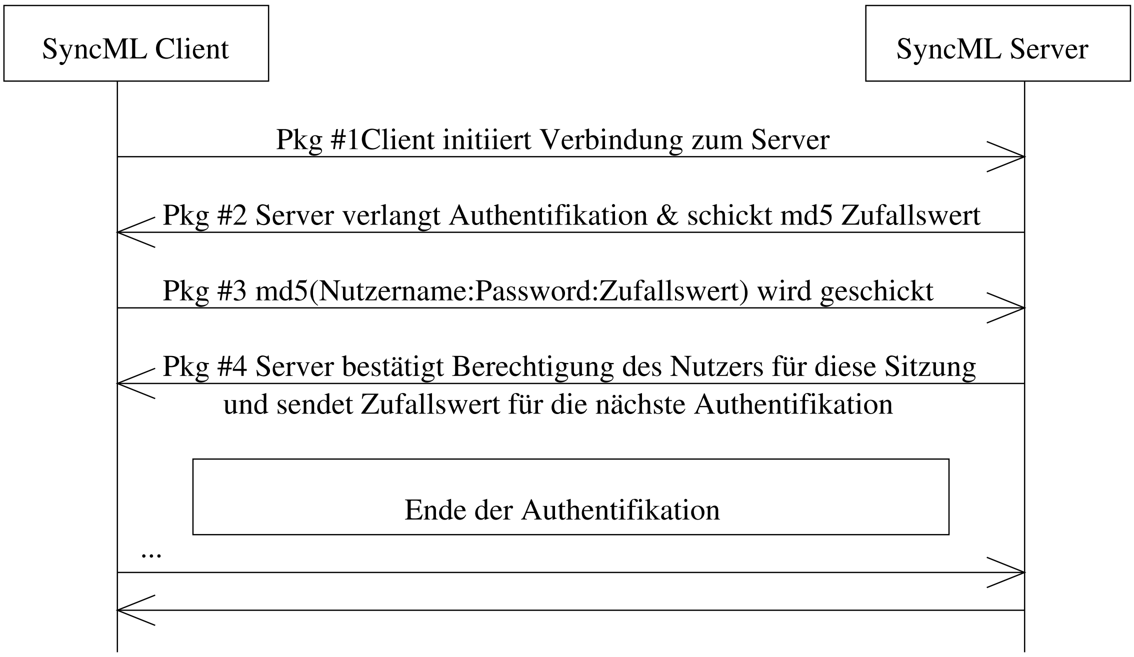


Abbildung 4.5: Paketaustausch bei MD5 Authentifikation

Der Client kann sich bei der nächsten Synchronisation mit der MD5 Authentifikation im ersten Paket an den Server wenden, da ihm der Zufallswert den er dafür braucht übermittelt wurde.

4.3.3 Verbindung Initialisieren

Die Authentifikation ist zwar bereits ein erster Initialisierungsschritt, aber da es dabei um einen Identitätsnachweis geht der nichts mit der eigentlichen Initialisierung der SyncML Synchronisation zu tun hat, wurde dieser Schritt separat behandelt.

Zur Initialisierung der Verbindung wendet sich der Client an den Server. Der Server kann dem Clienten aber per "Sync Alert" mitteilen, das dieser einen bestimmten Typ von Synchronisation einleiten soll. Bedingung dafür ist, das der Client erreichbar ist. Davon kann in den meisten Fällen nicht ausgegangen werden. Viele PDA´s werden nur zur Synchronisation an das Netzwerk angeschlossen und auch Mobiltelefone können abgeschaltet sein oder keinen Empfang haben. Das ist auch der Grund dafür, das der Server die Synchronisation nicht selbst einleitet, sondern den Clienten darum bittet.

Wenn der Client sich an den Server wendet, muss er das Target Element verwenden um das gewünschte Gerät und den darauf laufenden Service zu adressieren. Er muss sich ausserdem selbst per Source Element identifizieren. Das dient hier im Unterschied zur Authentifikation dazu, die Datenbanken zu adressieren. Target und Source werden gemeinsam mit dem zugehörigen Sync Anker in einem Alert Element übertragen. Auch die Art der Synchronisation (z.B. Zwei-Wege-Synchronisation) wird hier mit angegeben. Sollen mehrere Datenbanken synchronisiert werden, müssen auch mehrere Alert Elemente in einem Paket geschickt werden. Mit einem "put" Befehl werden die Eigenschaften des verwendeten Gerätes an den Server übermittelt. Das betrifft den Typ des Gerätes und die Anzahl und Art der noch speicherbaren Einträge.

Der Server muss seine Fähigkeiten daraufhin dem Clienten mitteilen, wenn dieser die Information anfordert. Der Server kann das aber auch tun, wenn die Information nicht angefordert wird. Er muss ausserdem unter Benennung von Quelle und Ziel die vom Clienten angebotene Art der Synchronisation akzeptieren, um diese zu ermöglichen. Was beim Clienten Quelle war ist nun Ziel und umgekehrt. Der Sync Anker des Servers wird mit an den Clienten übermittelt.

Wenn der Client nach dem Senden der Initialisierung keine Antwort vom Server erhält, muss er davon ausgehen das sein Paket nicht angekommen ist und es später noch einmal senden. Wenn der Server keine Antwort auf sein Initialisierungspaket zum Clienten erhält, dann darf er die Synchronisation abbrechen. Der Client muss dann zu einem späteren Zeitpunkt die Initialisierung von vorn beginnen.

4.3.4 Synchronisation

In Abschnitt 4.2.1 wurden alle möglichen Synchronisationstypen beschrieben. Nach der Initialisierung unterscheidet sich nur noch, wie synchronisiert wird. Wer die Synchronisation eingeleitet hat, spielt hier keine Rolle mehr. Die Anzahl der Synchronisationstypen die bei der Implementierung unterschieden werden müssen, lässt sich deshalb auf folgende begrenzen :

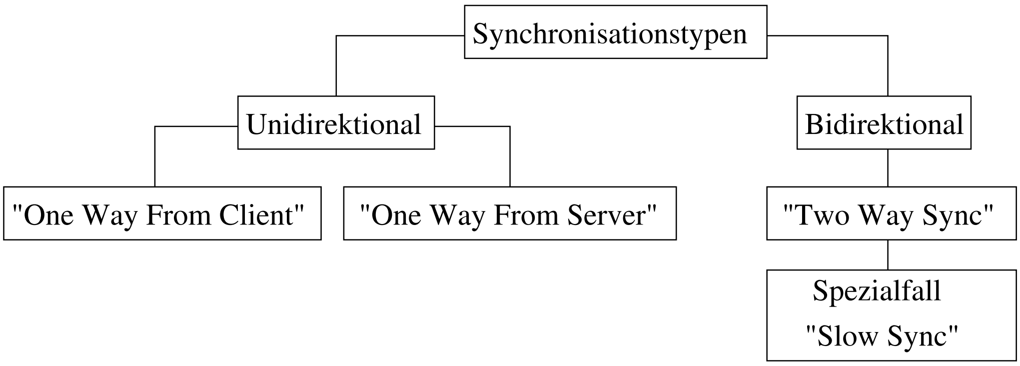


Abbildung 4.6: SyncML Synchronisationstypen ohne Unterscheidung des Initiators

Bei “Slow-Sync”, dem Spezialfall der Zwei-Wege-Synchronisation, werden alle Datenbankeinträge von Client und Server in der Sync-Engine des Servers verglichen, um an Hand dieser Daten Client- und Serverdatenbanken zu aktualisieren.

Wenn eine “One Way ..” Synchronisation verwendet wird, gibt es keinen Modus bei dem komplette Datenbanken in die des Kommunikationspartners eingearbeitet werden. Das liegt daran, das der Client über keine Sync-Engine verfügt und die nötigen Vergleiche nicht ausführen kann.

Client und Server können bei einer “One Way ..” Synchronisation komplette Datenbanken an den Kommunikationspartner senden. Die Daten werden beim Empfänger aber nicht mit existierenden Daten verglichen, sondern der Empfänger ersetzt mit diesen Daten die eigene Datenbank. Das entspricht dem Einspielen eines Backup. Dieser Vorgang ist zwar in beide Richtungen möglich, sollte aber in der Praxis nur vom Clienten in Anspruch genommen werden, da der Server auch bei einer einfachen “One Way From Client” Synchronisation die benötigten Informationen erhält.

4.3.5 WBXML

WBXML ist ein Binärformat für WML (Wireless Markup Language), wobei WML wiederum das Hypertext Format von WAP (Wireless Application Protocol) ist. Bei WBXML werden die für den Menschen gut lesbaren Bezeichner für Elemente und Attribute in kurze Bezeichner von ein Byte Länge umgesetzt. Durch diese Umsetzung fallen die zu übertragenden Nachrichten kürzer aus, was Zeit und Geld spart. Das Verhältnis von Protokoll- zu Nutzdaten wird dadurch verbessert und das führt bei den im Mobilfunkbereich verwendeten 9600 Bit/s zu erheblichen Einsparungen. Bei der Umsetzung nach WBXML wird auch die in SyncML vorhandene Redundanz genutzt, um sparsam mit den kurzen Bezeichnern umzugehen. So steht bei SyncML am Anfang eines Blockes ein Bezeichner und am Ende des Blockes ein “/” gefolgt von dem gleichen Bezeichner. Sprachelemente, die einen Block schliessen, haben deshalb bei WBXML keine eigenen Kurzformen, sondern werden durch “01” für Block schliessen, abgekürzt. Es wird also nur “/” gesendet, weil sich die Bedeutung aus der Struktur der Daten ergibt.

<Syncml>	6D
<SyncHdr>	6C
..	..
</SyncHdr>	01
<SyncBody>	6B
..	..
</SyncBody>	01
</Syncml>	01

Tabelle 4.3: Beispielumsetzung SyncML nach WBXML

4.4 Test auf SyncML Tauglichkeit

Auf dem Weg zu einem Synchronisationsstandard muss geprüft werden, ob er von Produkten, die auf der Basis dieses Standards arbeiten, eingehalten wird. Der Test auf SyncML Tauglichkeit oder Kompatibilität wird in zwei Schritten vollzogen.

Der erste Schritt ist das Ausfüllen eines Fragebogens. Hier wird unterschieden, ob es sich bei dem Produkt um einen Client oder einen Server handelt. Der Fragebogen unterteilt sich in die Abschnitte Geräteinformation, Client / Server Konformität, und Transport Konformität.

Die folgende Tabelle ist ein Beispiel für einen Auszug aus der Tabelle für die Befehlselemente des Representation Protokoll. Die Worte MUSS, SOLLTE, KANN sind zu verwenden wie in RFC 2119 beschrieben. Eingetragen werden auch Eigenschaften die zum Bestehen des Test nicht nötig sind. Bei Clienten mit wenig Resourcen wird aber meist nur implementiert, was unbedingt erforderlich ist. Die SyncML Protokolle enthalten ausreichend Redundanz, um bestimmte Befehle durch einen oder mehrere andere Befehle zu ersetzen. Der Client zur unten stehenden Tabelle hat z.B. keinen Add Befehl, um auf Serverseite Daten hinzuzufügen. Er kann das aber mit Replace tun, weil das Ersetzen nicht vorhandener Daten wie ein Hinzufügen zu behandeln ist.

	Benötigt	im Client	Implementiert	im Client
Befehl	Senden	Empfangen	Senden	Empfangen
Add	SOLLTE	MUSS	Nein	JA
Search	KANN	KANN	NEIN	NEIN
Replace	MUSS	MUSS	JA	JA
...	...	...	...	...

Tabelle 4.4: Auszug aus einer SyncML Konformitätserklärung

Nur wenn das Produkt laut erstem Schritt SyncML konform ist, wird es zum zweiten Schritt, einem Test auf Interoperabilität, zugelassen. Dieser Interoperabilitätstest findet auf einem SyncFest statt und wird vom SyncML Interoperability Committee (SIC) überwacht. Das SIC setzt sich aus Repräsentanten der Sponsorfirmen zusammen.

Ein solches SyncFest ist ein Ereignis auf dem sich die Entwickler von SyncML Produkten treffen. Diese Ereignisse finden in Abständen von einigen Monaten statt. Dort wird nachgewiesen, das ein Produkt mit den Produkten von mindestens zwei anderen Entwicklern Daten synchronisieren kann. Die Betonung liegt hier auf mindestens zwei anderen Produkten, denn der Zeitplan auf einem SyncFest ist gut gefüllt. Es wird versucht, mit so vielen Produkten wie möglich zu synchronisieren. Der einzige akzeptierte Grund den Test ohne Synchronisation abzubrechen, ist die Verwendung von Datenobjekten die kein anderes Produkt unterstützt.

Wenn das SIC bestätigt, das der Test erfolgreich verlaufen ist, darf der Entwickler das SyncML Logo im Zusammenhang mit dem Produkt verwenden.

Abbildung 4.7: SyncML Logo

4.5 Vor- und Nachteile

Vor- und Nachteile einer Sache sind natürlich immer vom jeweiligen Standpunkt abhängig. Was des einen Freud, ist des anderen Leid. Es gilt also, zuerst einen Standpunkt zu klären und unter diesem dann das Für und Wieder der Standardisierungsbestrebung im Bereich der Synchronisation von Daten zu erläutern.

Mir geht es darum, die Eigenschaften von SyncML aus der Sicht eines Anwenders zu betrachten, der bestrebt ist, die bei ihm nötige Datensynchronisation zu vereinfachen. Wenn der Wunsch des Anwenders Befehl für den Softwareentwickler ist, dann ist das auch dessen Sicht. Sollte der Softwareentwickler seine Nische aber gerade dadurch gefunden haben das Datensynchronisation unter Umständen schwierig bis unmöglich ist, dann hat er bestimmt eine andere Meinung zu diesem Thema.

Eine Standardisierung in diesem Bereich ist natürlich zu begrüssen. Der Anwender wird dadurch in die Lage versetzt, Daten zwischen vielen Applikationen auf unterschiedlichen

Geräte auszutauschen. Das sind Applikationen, die ohne den Standard bezüglich der Datenbasen Insellösungen darstellen oder nur mit wenigen Applikationen auf direkt verbundenen Geräten kommunizieren können. Und das, obwohl die verwendeten Datenformate oft weit verbreitet sind.

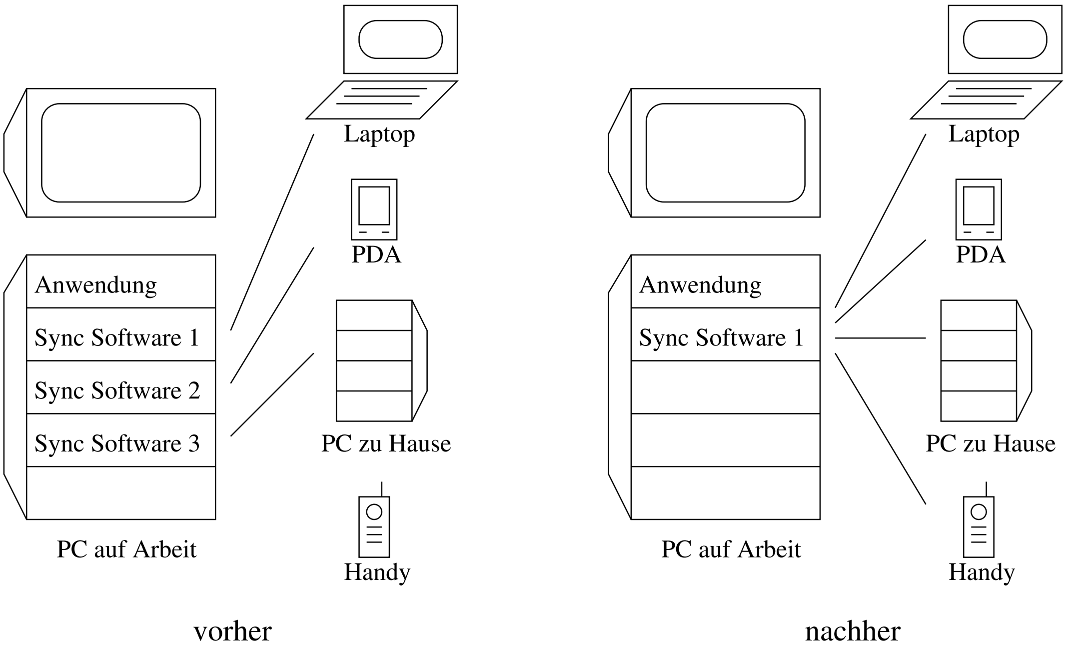


Abbildung 4.8: Höhere Verfügbarkeit durch Vereinheitlichung

Mit dem Standard kann die Synchronisation zu unterschiedlichen Geräten mit der selben Software erfolgen. Die Anzahl der zur Synchronisation eingesetzten Programme kann sich daher langfristig verringern. Die Anzahl wird sich erst deshalb langfristig verringern, weil nicht jedes Produkt sofort über den neuen Standard verfügt. Viele Geräte lassen sich diesbezüglich überhaupt nicht erweitern. Das macht, bis die Geräte ausser Dienst geststellt werden, den Einsatz der alten Software erforderlich.

Die Anzahl der Synchronisationen pro Zeit wird sich dagegen voraussichtlich erhöhen. Ich möchte das damit begründen, das der Wert der Daten im Vergleich zur Hardware, auf der sie bearbeitet werden, steigt. Es wird mehr Hardware eingesetzt. Und es ist auf dieser Hardware möglich, die Daten mit anderen Geräten zu synchronisieren.

Fazit: Wer in Zukunft mit SyncML Unterstützung genausoviel Daten synchonisieren möchte wie bisher, wird das mit weniger Aufwand tun können. Steigendes Datenaufkommen das synchronisiert werden muss, kann bewältigt werden, ohne gegenüber dem jetzigen Stand den Aufwand zu erhöhen.

Ein Standard hat aber nicht nur Vorteile, so bringt z.B. die Überwachung der Einhaltung Probleme mit sich. Die SyncML Initiative wird von ihren Mitgliedern gesponsort. Es handelt sich hierbei um gleichberechtigte Partner, die das Projekt entsprechend ihrer zur

Verfügung stehenden Resourcen unterstützen. Es wird also mit vielen Stimmen gesprochen und hier bei der Entscheidungsfindung einen Konsens zu erreichen, ist mindestens zeitaufwändig. Das die SyncML Initiative keine Führung hat, wird von der Gartner Group als Problem angesehen [29]. An dieser Stelle wird auch bemängelt, das einige Unternehmen mit Schlüsselpositionen auf dem Gebiet der PIM nicht an der Initiative teilnehmen.

Die Höhe der von den Gründungsunternehmen eingebrachten Resourcen kann entweder nicht schnell genug in die Struktur der Organisation umgesetzt werden, oder die Mittel sind zu knapp bemessen. Das ist an sich nichts Ungewöhnliches und ein Problem, mit dem viele Initiativen zu kämpfen haben. Nichtsdestoweniger wird durch diesen Flaschenhals die Entwicklung gebremst. Die Zahl der unterstützenden Unternehmen ist binnen eines Jahres von Null auf fast 700 gewachsen, was einem Überrennen gleich kommt. In der Mailingliste war zu erfahren, das Anfragen zur Teilnahme an einem SyncFest teilweise nicht beantwortet wurden. Weil neben den grossen Gründungsmitgliedern besonders kleiner Unternehemen auf diesem Gebiet ihre Chance sehen, ist es aber wichtig, bei der Etablierung von Produkten möglichst wenig Steine im Weg zu haben.

Trotz aller Probleme, welche die Initiative zu bewältigen hat, ist sie ein Schritt in die richtige Richtung, denn das Ziel ist nicht die Synchronisation von Daten, sondern das Ziel ist es synchronisierte Datenbestände zu haben. Und auf diesem Weg ist eine Vereinheitlichung sicher eine Hilfe.

Kapitel 5

Analyse bestehender Tools zur Implementierung

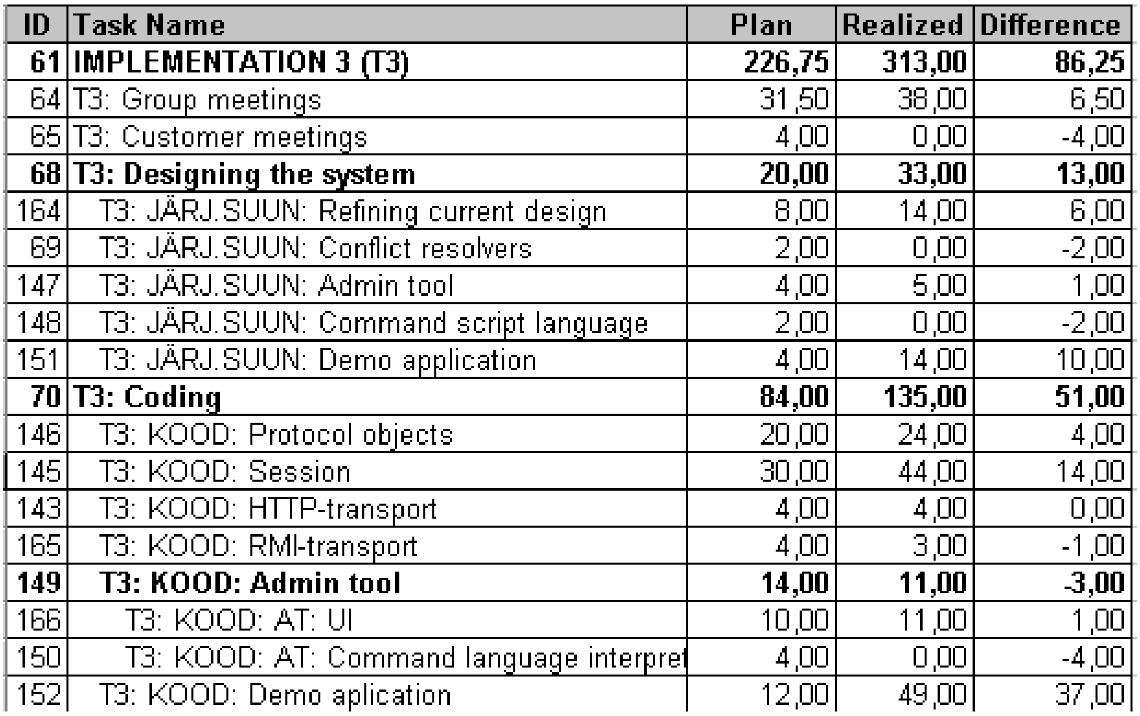
Bevor ich mit den Ausführungen zu einem eigenen Prototypen beginne, möchte ich zusammenfassen, was die Recherche über bereits existierende oder geplante Hilfsmittel ergeben hat. Von Interesse waren für mich die Fakten von wem die Software für wen geschrieben wurde, welche Programmiersprachen zum Einsatz kommen und welche Zielsysteme für die Implementierung vorgesehen sind.

5.1 SmartSync

SmartSync ist eine SyncML Implementierung, die im Rahmen eines Softwareprojektes der Helsinki University of Technology entstand. SmartSync ist kein eigenständiges Softwareprojekt, sondern ein Modul für die Smartner-Engine der Firma Smartner Information Systems Ltd. [23]. Für den Einsatz in anderen Applikationen steht dieses Modul nicht zur Verfügung, ich hätte also auch nicht damit arbeiten können. Die Angaben über den Aufwand bei der Entwicklung eines solchen Projektes fand ich jedoch sehr interessant.

Programmiert ist SmartSync in Java. Es wird eine Bibliothek für die Clienten- und die Serverseite bereitgestellt. Der Server enthält eine Sync-Engine zur Konfliktbehandlung. Für administrative Zwecke steht eine GUI Applikation zur Verfügung. Dem Clienten wird vom Server ein Teil der Arbeit abgenommen. Dennoch wird bei den Leistungsanforderungen von grösseren Resourcen ausgegangen als bei anderen Implementierungen, es wird ein Laptop vorausgesetzt.

Das Projekt wurde in 6 Phasen aufgeteilt. Planung, Implementierung (1-4) und Auslieferung. Die Umsetzung erfolgte mit sieben Leuten innerhalb von zwei Monaten. Ich habe die folgende Abbildung hier aufgenommen, weil sie mittels Daten aus der Praxis zeigt, das sich bei einem solchen Projekt der zeitliche Ablauf nur bis zu einem bestimmten Grad planen lässt.



ID	Task Name	Plan	Realized	Difference
61	**IMPLEMENTATION 3 (T3)**	**226,75**	**313,00**	**86,25**
64	T3: Group meetings	31,50	38,00	6,50
65	T3: Customer meetings	4,00	0,00	-4,00
68	**T3: Designing the system**	**20,00**	**33,00**	**13,00**
164	T3: JÄRJ.SUUN: Refining current design	8,00	14,00	6,00
69	T3: JÄRJ.SUUN: Conflict resolvers	2,00	0,00	-2,00
147	T3: JÄRJ.SUUN: Admin tool	4,00	5,00	1,00
148	T3: JÄRJ.SUUN: Command script language	2,00	0,00	-2,00
151	T3: JÄRJ.SUUN: Demo application	4,00	14,00	10,00
70	**T3: Coding**	**84,00**	**135,00**	**51,00**
146	T3: KOOD: Protocol objects	20,00	24,00	4,00
145	T3: KOOD: Session	30,00	44,00	14,00
143	T3: KOOD: HTTP-transport	4,00	4,00	0,00
165	T3: KOOD: RMI-transport	4,00	3,00	-1,00
149	**T3: KOOD: Admin tool**	**14,00**	**11,00**	**-3,00**
166	T3: KOOD: AT: UI	10,00	11,00	1,00
150	T3: KOOD: AT: Command language interpret	4,00	0,00	-4,00
152	T3: KOOD: Demo aplication	12,00	49,00	37,00

Abbildung 5.1: Auszug aus dem Zeitplan bei SmartSync; Angaben rechts in Stunden

Die Differenzen treten in den verschiedenen Projektstadien an ganz unterschiedlichen Stellen auf und ohne mehr Detailkenntniss über das Projekt lässt sich hierzu keine Wertung vornehmen. Das ist auch nicht mein Anliegen, für mich war es von grösserem Interesse, die Gewichtung der einzelnen Arbeitsvorgänge zu sehen. Auch wenn hier der grösste Teil der Literaturrecherche bereits hinter mir liegt, kann ich dadurch Schlüsse für die weitere Vorgehensweise ziehen. Für mich waren die unter [24] zu findenden Informationen zum Projektmanagement aus logistischer Sicht nützlich. Inhaltlich spielt SmartSync auf dem Weg zu meiner SyncML Prototyp Applikation keine Rolle.

5.2 SyncML Reference Toolkit

Das SyncML Reference Toolkit wird von der SyncML Initiative bereitgestellt und kann von der selben Seite geladen werden wie die SyncML Protokolle [21]. Das Toolkit steht in zwei Varianten zur Verfügung. Die erste ist für 32Bit Intel Architekturen unter Windows32 und Linux, sowie für PalmOS ab Version 3.5 geschrieben. Bei allen Hardwarearchitekturen kommt der gleiche Quelltext zum Einsatz, lediglich die Datentypen und Bibliotheksfunktionen wurden an das jeweilige Sytem angepasst. Die zweite Variante ist ausschliesslich für EPOC 6. Hier wurden die für dieses System relevanten Dateien zusammengefasst. Im Verzeichnisbaum des Handbuches [21] ist EPOC genau so eingegliedert wie palm, win und linux. Die Bereitstellung einer Version, die nur für EPOC bestimmt ist, hängt also nicht unmittelbar mit dem Toolkit selbst zusammen. Die PalmOS Variante des Toolkits muss mittels Crosscompiler auf einer anderen Plattform compiliert werden, bei den anderen Zielplattformen wird dort compiliert, wo das Toolkit eingesetzt wird. Da EPOC Geräte, von den Geräten auf denen compiliert wird, die wenigsten Systemresourcen haben, könnte die dafür optimierte Variante bereitgestellt worden sein.

Die folgende Grafik soll die Architektur des Toolkit veranschaulichen. Die Applikation kommuniziert mit der SyncML Implementierung und dem Transporthandler über getrennte API´s. Der Transporthandler ist in dieser Realisierung also kein Bestandteil der SyncML Implementierung und kann daher durch ein anderes Produkt ersetzt werden. Das kann sich in der Zukunft beim Transport über neue Kommunikationswege als sinnvoll erweisen.

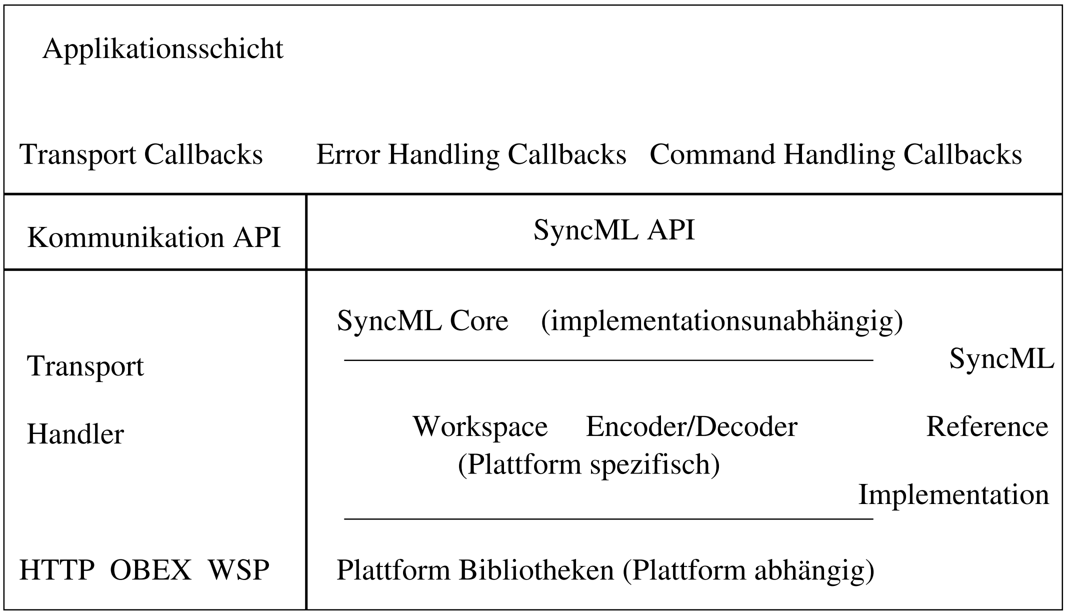


Abbildung 5.2: Architektur der SyncML Reference Implementation

Das Toolkit ist in C++ geschrieben. Ich habe es unter Linux mit dem gcc compiliert und 5 Objektdateien erhalten. In einer dieser Dateien sind die Funktionen zur SyncML Reference Implementation. Die anderen Dateien gehören zum Transporthandler und beinhalten Funktionen zur Übertragung mittels HTTP und WSP.

Der Aufwand für die Erstellung einer SyncML Anwendung mit Hilfe dieses Toolkit ist trotz der bereitgestellten Funktionalität nicht unerheblich. Das liegt daran, das dieses Toolkit in erster Linie eine Umsetzung der SyncML Protokolle ist und daher dichter an den Protokollen als an der Anwendung angesiedelt wurde. Für Anwendungen, die implementiert werden, ist das dahingehend von Vorteil, das mit dem Toolkit keine Richtung vorgegeben wird, in die zu implementieren ist. Also steht trotz Toolkit das gesamte Spektrum zur Verfügung, welches mit den Protokollen abdeckbar ist. Firmen,die Software entwickeln, sind die Zielgruppe des Toolkit. Diese Firmen haben so die Möglichkeit, Anwendungen für verschiedenste Einsatzgebiete des Datenaustausches zu schreiben und können trotzdem auf eine einheitliche Basis bei der Implementierung zurückgreifen.

Das Firmen die Zielgruppe sind, fällt auf der Projekthomepage sofort ins Auge. Es wird viel Wert darauf gelegt, wieviele und welche Unternehmen an der SyncML Initiative beteiligt sind. So ist es nicht verwunderlich, das ein Teil der Unterlagen nur durch registrierte Supporter eingesehen werden kann. Ich habe keine Möglichkeit, die dort untergebrachten Dokumente zu lesen. Daher ich auch nicht sagen, was sich dort alles befindet, aber über die Mailingliste zum SyncML Protokoll [22] war zu erfahren, das im SyncML Supporter-Bereich auch Beispielcode liegen soll.

Das Toolkit wurde am 06. März 2001 bereitgestellt und ist seit diesem Tag unverändert abrufbar. Zwischen dem Erscheinen von Version 1.0 der SyncML Protokolle und dem Toolkit ist eine Spanne von nur drei Monaten vergangen. Zum jetzigen Zeitpunkt steht das Toolkit bereits mehr als ein halbes Jahr unverändert zur Verfügung. Es ist also nicht zu erwarten, das in nächster Zeit eine verbesserte Version angeboten wird. Über die Homepage von SyncML sind diesbezüglich keine Informationen zu erfahren. Das gilt auch für das, in HTML verfügbare, Handbuch zum Toolkit. Gemessen am frühen Zeitpunkt der Erscheinung, enthält es eine gute Beschreibung der Funktionen. Das bezieht sich nicht nur auf das “wie”, sondern auch auf das “wann” des Einsatzes der Funktionen. Was die in den Funktionen verwendeten Datenstrukturen angeht, hilft das Handbuch kaum. Spätestens an dieser Stelle ist dann Arbeit am Quelltext nötig.

5.3 sync4j

Bei sync4j handelt es sich um eine Java Klassenbibliothek, die sich derzeit in einem Entwicklungsstadium zwischen Planung und Pre-Alpha befindet. Entwickelt wird dieses Projekt von Sean Sullivan und entstand im Unterschied zu den vorangegangenen Projekten als Initiative einer Einzelperson.

Trotz des frühen Entwicklungsstadiums ist es möglich, mit sync4j SyncML Dokumente zu generieren. Auch eine Transportbindung ist bereits implementiert. Ob und wann neben http weitere Transportmöglichkeiten bereitgestellt werden, ist noch nicht bekannt.

sync4j.client	Einstellung des Synctyps
sync4j.core	Fehlerbehandlung; SyncML Protokolle
sync4j.framework.client	Schnittstelle Kommunikation
sync4j.transport.http.client	http Bindung für Schnittstelle
sync4j.framework.server	Schnittstelle Kommunikation
sync4j.transport.http.server	http Bindung für Schnittstelle
sync4j.tests	Test auf Gültigkeit: Message, Header, Body, ..

Tabelle 5.1: sync4j Packages

Seit Oktober 2001 ist im Bereich der Projekthomepage [26] auch eine API Beschreibung für sync4j zu finden. Leider steht bisher nur bei Funktionen aus dem sync4j.core Packet beschrieben, welche Aufgabe diese haben. Für alle Pakete ist jedoch die eigentliche Schnittstelle beschrieben. Dabei wird auch auf Inhalte der verwendeten Datentypen eingegangen. Für die Entwicklung eines eigenen Prototypen hatte sync4j zu spät die erforderliche Funktionalität. Inzwischen lassen sich damit wohl erste Client Server Lösungen realisieren, allerdings ist dabei noch mit Schwierigkeiten zu rechnen. Anderenfalls wäre der Status des Projektes bestimmt nicht mehr Pre-Alpha.

5.4 Wireless Java Programming with J2ME

Hierbei handelt es sich um kein Softwareprojekt, das eine Funktionsbibliothek zur Verfügung stellt, sondern um ein Buch. In diesem Buch geht es um die Programmierung mit der Java 2 Micro Edition. Ein besonderer Gesichtspunkt ist, das die Geräte für die programmiert wird nicht “drahtgebunden” sind. Das bezieht sich neben der Netzwerkanbindung auf die verwendeten Resourcen, die gerade in mobilen Geräten knapp bemessen sind.

Das Buch teilt sich in zwei Teile, im ersten Teil geht es allgemein um die Programmierung in Java. Auch der Anfang des zweiten Teils ist bezüglich der Datensynchronisation mit SyncML noch nicht so interessant. Kapitel 10 handelt von der Verwendung von XML und hier wird nicht nur beschrieben wie und wo XML eingesetzt wird, sondern auch welche Arten von XML Parsern es gibt. Im Zusammenhang mit den Eigenschaften von Parsern mit Ereignis-basierter Schnittstelle und solchen mit Baumstruktur werden auch konkret Vertreter benannt und die Einbindung in Java gezeigt.

Eigenschaften	Ereignisbasiert	Baumstruktur
Proprietär	TinyXML; Ælfred	TinyXML; NanoXML
SAX Interface	NanoXML; Ælfred	

Tabelle 5.2: Vorgestellte Parser und deren Eigenschaften

Neben den Fähigkeiten der Parser hat auch hier wieder der Resourcenverbrauch einen hohen Stellenwert. Bei Grössen bis zu 30kByte wird es bestimmt Systemkomponenten mit einem höheren Speicherbedarf geben. Die längeren Laufzeiten bei einer Baumstruktur können aber schon Einfluss auf den verwendeten Parser haben.

Zu jedem Parser gibt es ein Beispielprogramm mit dem eine XML Datei geparst wird. Als XML Datei wird eine Mail und die zugehörige Document Type Definition (DTD) verwendet.

In der Tabelle oben wird SAX erwähnt. SAX ist eine Programmschnittstelle für ereignisbasierte XML Parser. Dadurch ist es möglich, den Parser zu tauschen ohne die Schnittstelle zu ändern.

In Kapitel 12 des Buches ist die Synchronisation eines Kalenders implementiert. Das Programm ist unter Verwendung des SyncML Representation Protocol realisiert. Die verwendeten Kalenderdaten müssen im vCalendar Format vorliegen. Intern arbeitet das Programm nicht mit dem vCalendar Format. Ein nicht unwesentlicher Teil des Implementierungsaufwandes besteht daher in der Umwandlung vom Internen in das vCalendar Format und umgekehrt.

Es wird davon ausgegangen, das ein mobiler Zeitplaner mit einer Netzwerkdatenbank abgeglichen wird. Dazu wird an beiden Kommunikationsendpunkten ein SyncAgent eingesetzt. In dieser Beispielimplementierung ist der Transport über http vorgesehen. Es ist nicht der komplette SyncML Sprachumfang implementiert, sondern eine Teilmenge, die nötig ist, um die gewünschte Funktionalität sicherzustellen. Dem Datenflussmodell ist zu entnehmen, das es sich bei der Synchronisation immer um eine vom Clienten initiierte Zwei-Wege Synchronisation handelt. Neben der Synchronisation wurde der Terminplaner noch mit einer Oberfläche versehen, in der es möglich ist, Termine hinzuzufügen, anzusehen und die Synchronisation auszulösen.

5.5 Geplante Projekte

5.5.1 LibSyncML

Über die Implementierung von LibSyncML [27] stehen bislang nur wenige Dinge fest. Als Programmiersprache soll C++ verwendet werden und das Projekt soll eine Bibliothek liefern, welche die Synchronisation übernimmt.

Das Projekt wurde von einem Studenten der TU-München ins Leben gerufen und wird dort vielleicht zu einer Diplomarbeit. Da zum jetzigen Zeitpunkt nicht mehr Informationen zu erhalten sind, bleibt mir für dieses Projekt nur, viel Erfolg zu wünschen. Eine freie Toolkitimplementierung, die dann vielleicht auch über einen langen Zeitraum gepflegt wird, würde die SyncML-Landschaft um einiges bereichern.

5.5.2 SyncML SDK

Das was in der Beschreibung zum SyncML SDK [28] geschrieben steht, ähnelt sehr stark dem von der SyncML Initiative bereitgestellten Toolkit. Das gilt für die Funktion, die Programmiersprache und die unterstützten Plattformen. Die Projekthomepage enthält seit Monaten nichts als einen Werbeslogan und drei nicht funktionierende Links. Das deutet mehr auf die Einstampfung des Projektes, als auf die Präsentation einer fertigen Version hin. Schade finde ich, das die Entwickler nicht mitteilen, was sie geplant haben. Mit dem, was zu erfahren ist, muss es sich um kein völlig neues Projekt handeln, sondern könnte auch eine Weiterentwicklung des Toolkit der SyncML Initiative sein.

Kapitel 6

Erstellung eines SyncML Prototyp

Der gedruckten Diplomarbeit ist eine CD beigelegt. Sollten Sie dieses Dokument auf elektronischem Wege erhalten haben, dann können Sie die Quellen, Beispiele und Hinweise unter http://www.misterm.de/diplom/index.html laden.

6.1 Funktionalität der angestrebten Lösung

Bei der angestrebten Lösung geht es in erster Linie darum, einen Datenabgleich mit Hilfe des neuen Standards SyncML zu implementieren. Ein zweiter Aspekt ist die Berücksichtigung des Koordinierungszwecks. Hierzu wurden zwei Szenarien genannt. Bei dem ersten möchte eine Person Daten auf verschiedenen Rechnern synchron halten, beim zweiten möchten verschiedene Personen gemeinsame Daten zum Zweck der Teamkoordination synchronisieren.

Für das erste Szenario gibt es viele Implementierungen. Die Synchronisation erfolgt oft mit proprietären Protokollen nur unter Berücksichtigung der benutzten Datenformate. Dieses Szenario ist ein Spezialfall des zweiten Szenario, denn natürlich kann eine Person von mehreren Rechnern aus Datensynchronisation mit dem Team betreiben. Wenn es bei diesen Daten um Kalenderdaten geht, dann sind anderen Teammitgliedern die Termine eines Einzelnen sicher egal. Interessant könnte aber sein, wann welche Person zur Verfügung steht, um daraus den Zeitpunkt und vielleicht Ort für ein Treffen abzuleiten. Es kommt also nicht nur auf die Synchronisation sondern auch auf die Betrachtungsweise der Daten an.

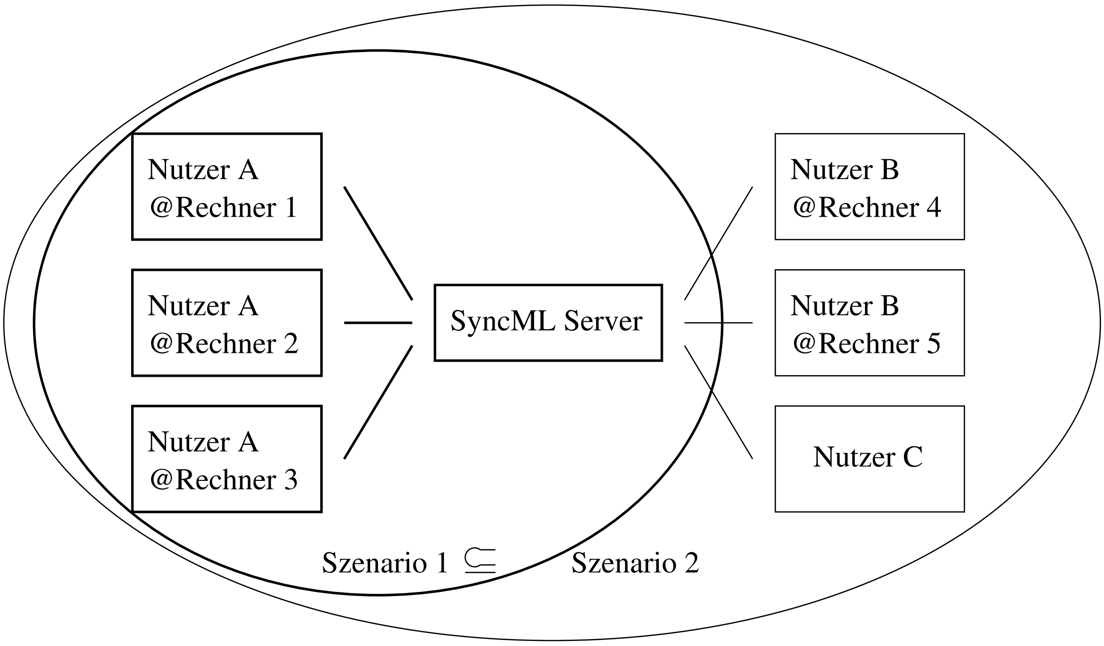


Abbildung 6.1: Zusammenhang zwischen Synchronisationsszenarien

Ich strebe daher eine Lösung an, die sowohl für eine einzelne Person mit verschiedenen Rechnern als auch für ein Team interessant ist. Vom Konzept her soll die Lösung dennoch einfach sein, damit sie in verschiedene Richtungen ausbaufähig und gut zu verstehen ist.

Als Beispiel das diesen Anforderungen gerecht wird, habe ich ein Gästebuch gewählt. Ein Client schickt eine vCard Visitenkartendatei an den Server. Der Server fügt die Datei seiner Datenbasis hinzu und stellt die Datenbasis zum Abruf bereit.

Ein einzelner Nutzer kann diese Implementierung von verschiedenen Rechnern aus als Adressbuch verwenden. Er kann Visitenkarten senden, entfernen und empfangen. Der Nutzer erhält durch die serverseitige Aufbereitung der Daten von vCard nach html die Möglichkeit, ohne seinen Clienten auf die Daten zuzugreifen. Diese zweite Schnittstelle für die Ausgabe der Daten eignet sich auch gut zu Kontrollzwecken, denn hier kann remote geprüft werden, ob Daten beim Server gültige Daten ankommen.

Für die Teamkoordination kann die Implementierung so eingesetzt werden, das die Teammitglieder die Visitenkarten aller am Projekt beteiligten Personen an den Server senden. Jeder sendet die ihm bekannten Visitenkarten und das ganze Team hat Zugriff auf die Liste.

Auch das Mischen zwischen Privat und Teamdaten ist möglich. Und das ohne das der Anwender Daten erhält, die ihn nicht interessieren oder er mit zwei Servern synchronisieren muss. Für eine Demonstration würde es reichen, wenn jeder Nutzer die Namen seiner privaten Dateien mit seinem Namen beginnen lässt, und die für das Team interessanten Daten z.B mit “Team”. Bei einem Datenabgleich muss er dann nur die Dateien, welche mit seinem Namen oder mit “Team” beginnen, synchronisieren. Bei einer praxistauglichen Realisierung für dieses Szenario ist die Respektierung der Privatsphäre sicherzustellen.

Implementieren werde ich sowohl den Client als auch den Server, denn zum gegenwärtigen Zeitpunkt gibt es keinen frei verfügbaren Server und einer der in Hardware gegossenen Clienten steht mir nicht zur Verfügung. Ausserdem ermöglicht die Implementierung beider Kommunikationspartner, Programmierern die sich mit dem RTK befassen möchten, einen leichteren Einstieg.

6.2 Wahl der Programmierhilfen

Mit Hilfe der gesammelten Informationen für das vorige Kapitel muss ich meine Entscheidung darüber treffen, womit ich programmieren werde. Zusatzbedingung ist die Verfügbarkeit des Tools und der damit verbundenen Programmiersprache für das Betriebssystem Linux auf x86 Architekturen. Grund für diese Bedingung ist, das ich auf meinem, aus dieser Konfiguration bestehendem System Root-Rechte habe, und ich dadurch die Installation für die Entwicklung benötigter Software selbst vornehmen kann.

sync4j befindet sich derzeit im Pre-Alpha Stadium. sync4j ist in Java programmiert, was Plattformunabhängigkeit schafft. Es lassen sich jetzt SyncML Dateien generieren. Ein Transport über http ist möglich. Zu dem Zeitpunkt als ich meine Entscheidung treffen musste funktionierte beides noch nicht. Eine Prototypentwicklung mit sync4j wäre also zum grossen Teil eine Entwicklung am Toolkit selbst gewesen. Deshalb verwende ich es nicht.

Universelle XML-Parser mit der DTD von SyncML zu verwenden, ist natürlich auch möglich. Das hat den Vorteil, das man erst die Programmiersprache wählen kann und sich dann nach einem geeigneten Parser umsieht. Aus Gründen des Resourcenverbrauches werden oft ereignis-basierte Parser eingesetzt. Die Callback-Funktion muss dann aber den sie betreffenden Kontext selbst ermitteln. Parser die eine Baumstruktur aufbauen, können Funktionen gezielter Informationen zukommen lassen und vermindern dadurch den Aufwand in der Funktion selbst. Also, wenn eine Implementierung mit einem XML-Parser, dann nach Möglichkeit mit Baumstruktur (z.B. DOM Schnittstelle).

Mit dem Parser wird aber nur das “Representation Protocol” umgesetzt. Zu SyncML gehört aber auch das “Sync Protocol”, in dem die Kommunikation zwischen Client und Server spezifiziert ist. Dieser Teil müsste ohne Verwendung eines Toolkit komplett neu implementiert werden.

Da ich, soviel will ich vorwegnehmen, ein akzeptables Toolkit gefunden habe, verzichte ich auf die Implementierung eines Prototypen auf dieser Ebene. Dadurch vermeide ich es die Fehler, die Andere gemacht und berichtigt haben, erneut zu begehen. Im Bezug auf die beiden SyncML Protokolle kann ich dafür eine vollständigere Lösung bieten. Ausserdem liefere ich bei Verwendung des Toolkit, das ich momentan für am weitesten fortgeschritten halte, eine Analyse über dessen Stärken und Schwächen.

Das Reference Toolkit (RTK) der SyncML Initiative ist mit Abstand das umfangreichste Tool, wenn es um SyncML Realisierungen geht. Das RTK ist in C++ implementiert und stellt sowohl Funktionalität für das "Reference Protocol", als auch für das "Sync Protocol" bereit. Grösserer Wert wurde auf die Erstellung von SyncML Dokumenten gelegt. Es werden XML und WBXML unterstützt und die Umschaltung erfolgt durch nur zwei Änderungen in der Anwendung. Mehr der Vollständigkeit wegen enthält das RTK auch HTTP und OBEX als Beispielimplementierung für das "Sync Protocol".

Ich habe mich trotz des hohen Einarbeitungsaufwandes für das Reference Toolkit entschieden. Das liegt daran, das es in absehbarer Zeit keine andere Implementierungshilfe mit diesem Funktionsumfang geben wird. Nachteilig finde ich, das nichts über eine Weiterentwicklung des RTK zu erfahren ist. Im Handbuch ist zwar von reservierten Funktionen für den zukünftigen Gebrauch zu lesen, aber von der SyncML Initiative wird das RTK wohl als fertig betrachtet.

Für die Entwicklung der Oberfläche des Clienten verwende ich TclTk. Mit dieser interpretativ abzuarbeitenden Scriptsprache lassen sich effektiv Benutzeroberflächen gestalten. Ausserdem ist eine Einbindung in c Programme möglich und das vereinfacht die Programmstruktur zugunsten der Übersichtlichkeit und Lesbarkeit des Quellcodes.

Die Programmierung erfolgt also in Linux mit c++ unter Verwendung des RTK und TclTk.

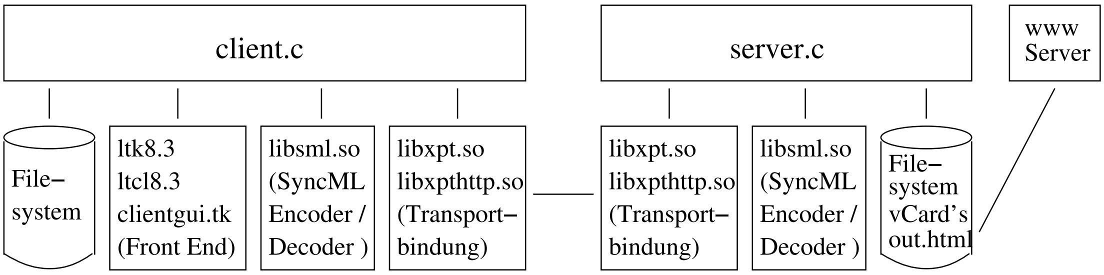


Abbildung 6.2: Schema der Prototyp-Implementierung

6.3 Client

6.3.1 Representation Protocol

Das Toolkit lies sich relativ unkompliziert compilieren. In einer Bibliothek ist eine Referenz nicht aufgelöst und die Bibliothek konnte aus diesem Grund nicht geladen werden. Die Funktion in der das Problem auftritt, wird aber, soweit ich feststellen konnte, nur in der Palm OS Version des Toolkit benötigt. Ich habe den Fehler berichtigt und für das erneute compilieren auch das Flag "-g" für Debuginformationen gesetzt. Weil die Datenstrukturen kaum dokumentiert sind, ist die Arbeit am Quelltext sehr wichtig.

Im Client werden zuerst globale Einstellungen zum Toolkit vorgenommen. Das betrifft die Grösse des Workspaces den alle Instanzen zusammen belegen dürfen und eine Ausgabefunktion, die wenn sie existiert, von Funktionen des Toolkit aufgerufen wird, um Statusinformationen auszugeben. Ich verwende nur eine SyncML Instanz, denn nach dem der Client sein Dokument an den Server übertragen hat, kann diese wiederverwendet werden. Die ausgegebenen Statusinformationen sollen bei der Programmierung behilflich sein. In einem fertigen Programm hat diese Funktion nichts mehr zu suchen. Um Fehler aufzuspüren, sind die Rückgabewerte der aufgerufenen Toolkit-Funktionen wichtiger. Die meisten Fehlercodes sind in der Datei smlerr.h dokumentiert. Diejenigen, die das Transportprotokoll betreffen, werden in der Datei xpt_error.txt berechnet.

```
#define SML_ERR_A_XPT_ERROR 0x5001
#define SML_ERR_A_XPT_SERVER_AUTH SML_ERR_A_XPT_ERROR + 0x02
        // Server authentication failed
```

Die Fehlermeldungen sind dadurch schwer zu finden, weil das grep Kommando an dieser Stelle scheitert. Von der erhaltenen Fehlermeldung 0x5001 abziehen zu müssen, um zu erfahren, wo der Fehler liegt, ist sicher auch nicht überall gängige Praxis.

Nachdem die globalen Einstellungen getroffen sind, wird eine Instanz angelegt. Hierbei wird die Grösse der durch sie belegten Speichers festgelegt, der natürlich kleiner sein muss als die vorher festgelegte Obergrenze. Die zweite wichtige Einstellung ist, ob mit XML oder WB_XML codiert werden soll. Beim decodieren wird beides verstanden. Es ist also möglich, das ein Kommunikationspartner XML und der andere WB_XML verwendet. Ich lasse, der besseren Lesbarkeit wegen, nach XML codieren.

Nun beginnt der Teil, in dem das SyncML Dokument generiert wird. Hier werden zuerst einige Daten festgelegt, die im Headers Platz finden (Versionsnummer, Protokolltyp, ..). Zum Body des Dokuments können jetzt SyncML Befehle hinzugefügt werden. Diese Befehle beziehen sich auf die Daten, die zum oder vom Server transferiert werden sollen. Ich habe keinen Mechanismus integriert, der festlegt, welche Daten abgeglichen werden müssen. Diese Aufgabe kann automatisiert werden, obliegt aber in meiner Realisierung dem Anwender. In Abhängigkeit der getroffenen Entscheidung wird jetzt das SyncML Dokument ergänzt. Wenn ausser dem Befehl auch Daten eingefügt werden müssen, geschieht auch das an dieser Stelle. Wenn dem Encoder im Toolkit jetzt noch mitgeteilt wird, das keine weiteren Nachrichten angefügt werden sollen, schliest er das Dokument ab. Die Applikation erhält dann Zugriff auf das Dokument und kann es an das Kommunikationstoolkit weiterreichen.

```
<?xml version="1.0" encoding="UTF-8"?><SyncML xmlns='SYNCML:SYNCML
1.0'><SyncHdr><VerDTD>1.0</VerDTD><VerProto>SyncML/1.0</VerProto><
SessionID>1</SessionID><MsgID>1000</MsgID><Target><LocURI>127.0.0.
1</LocURI></Target><Source><LocURI>vCardApplication</LocURI></Sour
ce><NoResp/></SyncHdr><SyncBody><Sync><CmdID>0-8/15</CmdID><NoResp
```

```
/><Add><CmdID>1</CmdID><NoResp/><Item><Source><LocURI>klaus.vcf</L
ocURI></Source><Data>begin:vcard n:Mustermann;Klaus
tel;cell:+49 (0) 177 991234
x-mozilla-html:FALSE
url:http://www.km.de org:Mustermann Inc.
adr:;;Musterstr. 88;Musterstadt;;88888; version:2.1
email;internet:Klaus.Mustermann@km.de
x-mozilla-cpt:;18912
fn:Klaus Mustermann
end:vcard </Data></Item></Add></Sync><Final/></SyncBody></SyncML>
```

In diesem Beispiel soll der Server das Datum mit der Bezeichnung “klaus.vcf” zu seiner Datenbasis addieren. Im Toolkit wird zwar neben der Übergabe der Daten auch oft deren Länge verlangt. Damit könnte ein StringCopy für binäre Daten realisiert werden. Zum kopieren von Zeichenketten wird aber strcpy verwendet und das setzt Null-terminierte Strings voraus. Laut Header ist sogar sicherzustellen, das die Nutzdaten UTF-8 konform sind.

6.3.2 Sync Protocol

Der Kommunikationsteil des RTK untestützt HTTP und WSP. Aus Gründen der vorhandenen Infrastruktur beschränke ich mich auf HTTP. Einer Applikation kann aber ein Transportprotokoll hinzugefügt werden, ohne sie neu compilieren zu müssen. Dazu sind die Namen der Bibliotheken, in denen die Transportprotokolle implementiert sind, in die Datei dynamicTransports einzufügen. Natürlich müssen die Bibliotheken bezüglich ihrer API den Anforderungen des Toolkit entsprechen. Die Datei dynamicTransports existiert im Toolkit nicht und ist auch nicht dokumentiert. Das sie angelegt werden muss, ergibt sich allein aus dem Quelltext des RTK.

```
mmr@pumuckl:~/syncml/src/bld/linux > ll *.so
-rwxr-xr-x 1 mmr users 244197 Jun 11 20:43 libsml.so
-rwxr-xr-x 1 mmr users 72878 Jun 11 20:43 libxpt.so
-rwxr-xr-x 1 mmr users 150908 Jun 11 20:43 libxpthttp.so
mmr@pumuckl:~/syncml/src/bld/linux > cat dynamicTransports
xpthttp
mmr@pumuckl:~/syncml/src/bld/linux >
```

Nachdem der Zeiger für die Position des generierten Dokumentes an die Applikation übergeben wurde, fängt diese an, die Kommunikation aufzubauen. Die Reihenfolge, in der die Funktionen des Kommunikations Toolkit aufgerufen werden und was ihr Aufruf bewirkt, ist unten beschrieben.

xptGetProtocols()

Anfordern einer Liste mit verfügbaren Transferprotokollen.

xptGetProtocol()

Anfordern von Informationen zu einem bestimmten Transferprotokoll.

xptSelectProtocol()

Auswahl eines zu verwendenden Transferprotokolls. Wenn sich an der Anzahl und Reihenfolge verfügbarer Transportprotokolle nichts ändert, kann auf xptGetProtocols() und xptGetProtocol() verzichtet und die Protokoll-ID direkt angegeben werden. Die Sender / Empfänger Rolle wird hier festgelegt.

xptOpenCommunication()

Verbindung zum Server wird aufgebaut. Ab diesem Zeitpunkt muss dieser empfangsbereit sein.

xptBeginExchange()

Austausch von Protokollinformationen, z.B. HTTP Informationen.

xptSetDocumentInfo()

Dokumenttyp, -länge, -name, .. setzen und senden.

xptSendData()

Dokumentdaten werden gesendet. Senden der Daten kann in mehreren Schritten erfolgen. Das ist sinnvoll, wenn die Speicherkapazitäten des Clienten nicht ausreichen, um das Dokument am Stück zu bearbeiten.

xptSendComplete()

Senden abschliessen, es folgt kein weiteres xptSendData().

xptGetDocumentInfo()*

Dokumentinfo vom Kommunikationspartner empfangen.

xptReceiveData()*

Dokument empfangen. Wenn der eigene Puffer kleiner ist als die Dokumentlänge oder weniger Daten empfangen werden als das Dokument lang ist, erneut xptReceiveData() aufrufen.

xptEndExchange()

Kein weiterer Austausch von Dokumenten.

xptCloseCommunication()

Verbindung wird geschlossen.

Die Reihenfolge der Funktionsaufrufe gilt für eine, vom Clienten initiierte, Kommunikation. Die mit * markierten Funktionen werden bei einer zwei-Wege Synchronisation benötigt, alle anderen bei ein- und zwei-Wege Synchronisation. Zum Verbindungsaufbau wird Adresse und Zielport des Servers benötigt. Diese Informationen sind vom Anwender einzugeben.

6.3.3 Front-End

Die Benutzeroberfläche des Clienten, sein Front-End, wird zur Eingabe logistischer Grössen benötigt. Hier wird festgelegt, welche Daten wohin synchronisiert werden sollen. Die benötigten Parameter im einzelnen :

action - Bezeichner für das auszuführende SyncML Kommando.

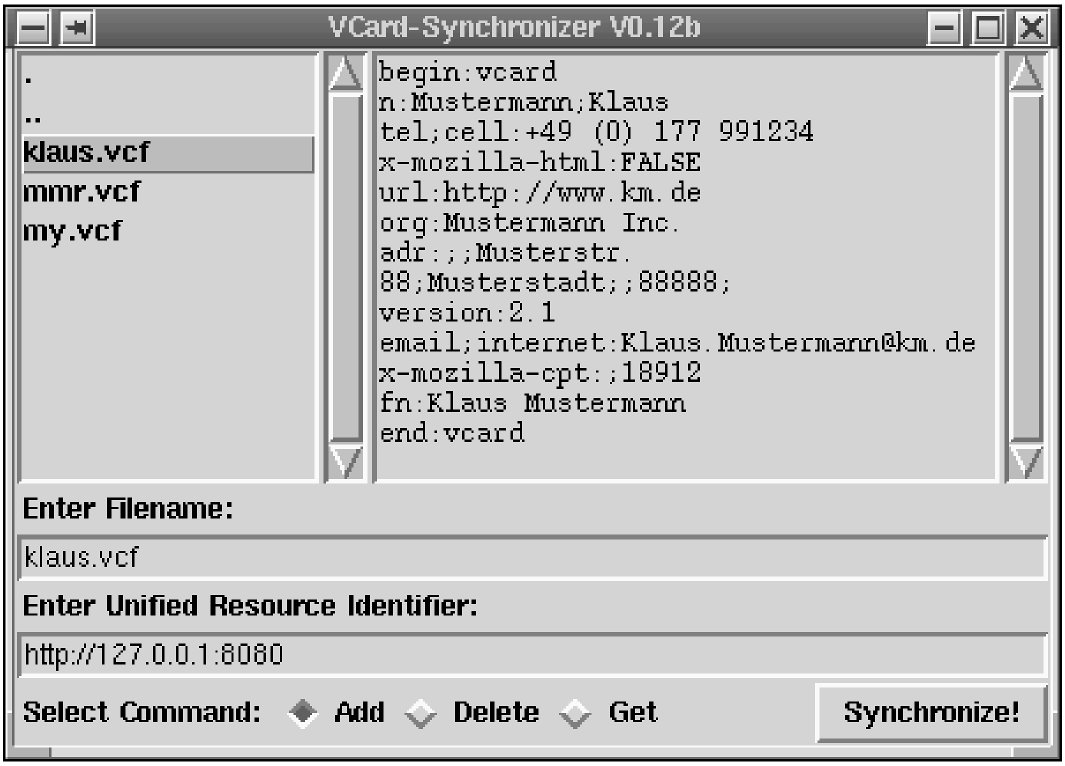
```
typedef enum {
  READY   =0,
  ADD     =1,
  DELETE  =2,
  GET     =3,
  REPLACE =4
} myCmdId_t;
```

Eine Variable dieses Typs kann, nachdem das SyncML Kommando hinzugefügt wurde, auf READY gesetzt werden. Das zeigt an, das ihr Inhalt ausgewertet wurde.

URI - Hier ist die Adresse des Servers einzutragen. Da ich nur http als Transportprotokoll verwende muss die URI immer mit http:// beginnen. Nach der IP Adresse ist, getrennt durch einen Doppelpunkt, die Portnummer des Servers anzugeben.

myfilename - Dateiname der zu synchronisierenden Datei. In Zusammenhang mit dem GET Befehl sind auch Angaben wie *.vcf oder *.* gültig.

mydata - Der Inhalt der zu synchronisierenden Datei. Bei den Befehlen DELETE und GET wird dieser Parameter nicht ausgewertet.



```
begin:vcard
n:Mustermann;Klaus
tel;cell:+49 (0) 177 991234
x-mozilla-html:FALSE
url:http://www.km.de
org:Mustermann Inc.
adr:;;Musterstr.
88;Musterstadt;;88888;
version:2.1
email;internet:Klaus.Mustermann@km.de
x-mozilla-cpt:;18912
fn:Klaus Mustermann
end:vcard
```

Enter Filename: klaus.vcf

Enter Unified Resource Identifier: http://127.0.0.1:8080

Select Command: ● Add ◇ Delete ◇ Get

Abbildung 6.3: graphische Oberfläche des Clienten

Um die Eingabe des Dateinamens zu erleichtern, habe ich in die Oberfläche einen File-Requester integriert. Die Navigation erfolgt durch Einzelklick mit der linken Maustaste. Bei Wahl eines Verzeichnisnamens wird dieses betreten. Bei Auswahl einer Datei wird ihr Name im Eingabefeld unter dem File-Requester angezeigt und der Dateiinhalt in den Editor rechts eingetragen.

Der Editor ist auch der Hauptgrund, warum ich eine graphische Oberfläche verwende. Natürlich ist der File-Requester eine Zugabe, mit der die Eingabe beschleunigt wird. Und auch die Möglichkeit zu haben, die Reihenfolge für die Parametereingabe selbst festlegen zu können, erscheint sinnvoll. Wichtigster Punkt ist jedoch, das eine Kopie der gewählten Visitenkarte im Editor bearbeitet werden kann, ohne das Original auf Festplatte zu beeinflussen. Damit kann die vollständige Visitenkarte auf den erforderlichen Inhalt zusammengekürzt werden.

Wenn die Visitenkarte an das Gästebuch eines Geschäftspartners gesendet wird, müssen private Telefonnummern sicher nicht enhalten sein. Auf der anderen Seite wirken Anrufe von Bekannten im Büro oft störend, so das bei ihnen der Geschäftsbereich gelöscht wird.

Diese Editorfunktion soll nur eine Anregung dafür sein, das es immer wichtiger wird, Synchronisationsprogramme mit Filterfunktionen auszustatten. Eine Unterteilung in Kategorien ist dazu ein sinnvoller Schritt. Ziel dieser Entwicklung ist es, das jeder die Informationen erhält, die er benötigt.

Im Eingabefeld für die URI steht beim Start des Clienten ein Beispieleintrag, um dem

Anwender das Format in dem die Eingabe verlangt wird, zu zeigen. Von den SyncML Befehlen, die am unteren Rand der Oberfläche geschrieben stehen, kann pro Synchronisation nur einer gewählt werden.

Es ist problemlos möglich, eine Taste für "Befehl hinzufügen" zu ergänzen. Die gewählten Befehle und zugehörigen Dateien könnten in Form von SyncML Befehlen dem Dokument hinzugefügt werden. Beim Klick auf "Synchronize!" würde das generierte Dokument an den Server geschickt. Wenn ich dann noch einen Mechanismus zum Prüfen der Daten auf Veränderungen hinzufüge, kann ein automatischer Datenabgleich erfolgen. Der Client würde dann mit einer einzigen Taste auskommen und auf dieser würde dann z.B. "Jetzt!" stehen. Das Prüfen auf Veränderungen im Datenbestand ist aber ein Fall für jemanden aus der Fachrichtung Datenbanken, während ich die Sychronisation mit Sicht auf einbezogene Netzwerke betrachte. Bei der Realisierung des Prototypen möchte ich dem Anwender die Möglichkeit geben, das Geschehen sehr elementar zu beeinflussen und stelle deshalb die Befehle Add, Delete und Get zur Verfügung.

6.4 Server

6.4.1 Aufruf

Wenn der Server gestartet wird, ist es wichtig ihm einige Parameter zu übergeben. Er muss den Port wissen, auf dem er die Anfrage eines Clienten erwarten soll. Dann verwende ich ein extra Verzeichnis zur Speicherung der vom Client empfangenen Visitenkartendateien. Der Name des Verzeichnis ist der zweite Parameter. Der Server schreibt das eigentliche Gästebuch im HTML Format in eine Datei, deren Name als dritter Parameter zu übergeben ist. Es ist sinnvoll hier eine Datei anzugeben, die von einem Webserver erreicht wird. Ein möglicher Aufruf ist:

```
x@y:~/xml/ > server 8080 testdir/ /usr/local/httpd/htdocs/guestbook.html
```

6.4.2 Sync Protocol

Der Server wartet auf den Kontakt durch einen Clienten und wertet anschliessend die Kommunikationsdaten aus. Es wird also im Server zuerst die Funktionalität des Kommunikations Toolkit benötigt. Über xptGetProtocols() und xptGetProtocol() wird ermittelt, welche Transportprotokolle zur Verfügung stehen. Die Existenz von Funktionsbibliotheken, die ein Transportprotokoll beinhalten, wird dem Server über den Inhalt der Datei dynamicTransports mitgeteilt. In dieser Datei ist, wie beim Clienten, nur die Bibliothek für den Transport über HTTP eingetragen. Beim Aufruf von xptSelectProtocol() kann daher auch nur dieses Protokoll, mit der Ergänzung, das die Anwendung Server ist, eingestellt werden.

Die eigentliche Verbindung erfolgt durch xptOpenCommunication(). Dieser Ruf wirkt blockierend. Die Funktion kehrt erst nach dem Verbindungsaufbau durch den Client zurück. Die Anwendung fragt dann Informationen über das einzulesende Dokument ab. Das ist vor allem für die Länge des Dokumentes wichtig, weil hier entschieden werden muss, ob am Stück eingelesen und ausgewertet werden kann. Falls das aus Speicherplatzgründen nicht möglich ist, kann lesen und auswerten in mehreren Durchläufen erfolgen.

```
xptGetProtocols()
xptGetProtocol()
xptSelectProtocol()
xptOpenCommunication()
    xptBeginExchange()
        xptGetDocumentInfo()
        xptReceiveData()
        xptSetDocumentInfo() *
        xptSendData() *
        xptSendComplete() *
    xptEndExchange()
xptCloseCommunication()
```

Tabelle 6.1: Reihenfolge der Funktionsaufrufe im Kommunikations Toolkit als Server

Bei der Verwendung dieser Funktionen im Server wird von einer Client initiierten Einweg-Synchronisation ausgegangen. Die mit * gekennzeichneten Funktionen werden bei einer Zwei-Wege Synchronisation zusätzlich zur Datenübertragung an den Clienten benötigt.

Der Speicherbereich, in dem das empfangene SyncML Dokument geschrieben wird, muss von der Anwendung bereitgestellt werden. Der Funktion xptReceiveData() wird nur ein Zeiger übergeben. Die eigentliche Reservierung erfolgt beim Anlegen eines SyncML Workspace, doch das ist Teil des “Representation Protocol”.

Um zu prüfen, ob und wenn ja, welche Informationen vom Client ankommen, habe ich einen TCP-Server implementiert, der Verbindungen akzeptiert und die übertragenen Informationen ausgibt. Dieser Testserver gehört nicht zur eigentlichen Prototypimplementierung, gibt aber die Möglichkeit Daten zu betrachten, die sonst vom Toolkit ausgewertet würden und die Anwendung nicht erreichen.

```
mmr@pumuckl:~/xml/source/by_mmr > testserver 8080
POST /my_doc.sml HTTP/1.1
Cache-Control: private
Connection: close
User-Agent: HTTP SyncML Client [en] (WinNT; I)
Accept: application/vnd.syncml-xml, application/vnd.syncml-wbxml, */*
Accept-Language: en
Accept-Charset: utf-8
Host: 127.0.0.1:8080
```

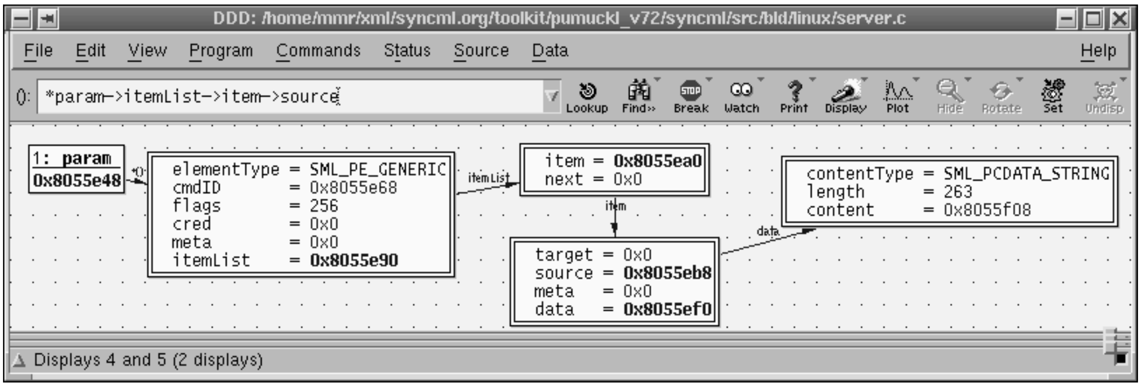
```
Content-Type: application/vnd.syncml-xml
Content-Length: 747
<?xml version="1.0" .. </SyncML>
```

Die Angaben für die HTTP Version, Cache-Control, User-Agent, akzeptierte Datentypen und Sprache sind in der Datei xpt-http.c des Toolkits fest verdrahtet. Eine Änderung kann nur im Quelltext durchgeführt werden. Sicher ist das ein Punkt der in künftigen Versionen, so es diese geben wird, von einer Funktion übernommen wird. Das es sich um einen Client unter WinNT handeln soll, ist dabei kaum das Problem. Die Akzeptanz von */* sollte aber durch Datentypen ersetzt werden, die der Empfänger wirklich versteht.

Wenn die Daten vom Client gelesen sind, bleibt die Verbindung noch bestehen. Erst nach dem Parsen des SyncML Dokument ist bekannt, ob der Client eine Antwort erwartet. Ob dem Clienten ein Dokument zurückgesandt wird, teilen die Callback-Funktionen über ein Flag mit, das nach dem Parsen ausgewertet wird.

6.4.3 Representation Protocol

Ähnlich wie beim Client ist das Toolkit auch beim Server zu initiieren. Das betrifft zuerst wieder die Grösse des Workspace und das Vorhandensein einer Ausgabefunktion. Auch der Encoder-Typ ist zu setzen. Da die Einstellung XML oder WB_XML in Abhängigkeit des vom Client gesendeten Dokumentes getroffen werden muss, empfiehlt es sich die Funktion smlInitInstance() nach xptGetDocumentInfo() aber vor xptReceiveData() aufzurufen. Weil ich aber sowohl im Client als auch im Server nur mit XML codierten Daten arbeite, rufe ich smlInitInstance() bei Programmstart dort auf, wo auch alle anderen Variablen initialisiert werden auf.



```
(): *param->itemList->item->source
```

```
1: param
0x8055e48
```

```
elementType = SML_PE_GENERIC
cmdID       = 0x8055e68
flags       = 256
cred        = 0x0
meta        = 0x0
itemList    = 0x8055e90
```

```
item = 0x8055ea0
next = 0x0
```

```
target = 0x0
source = 0x8055eb8
meta   = 0x0
data   = 0x8055ef0
```

```
contentType = SML_PCDATA_STRING
length      = 263
content     = 0x8055f08
```

```
Displays 4 and 5 (2 displays)
```

Abbildung 6.4: Anzeige von Datenstrukturen im Debugger

In smlInitInstance() werden dem Toolkit auch die Callback Funktionen mitgeteilt. Die Callbackfunktionen werden vom Toolkit beim Parsen des Dokumentes aufgerufen, um die im Dokument transportierten Nutzdaten zu handhaben. Einer aufgerufenen Callback Funktion werden Informationen zum Dokument übergeben, welches abgearbeitet wird. Es werden anwendungsspezifische Daten übergeben und natürlich die Daten aus dem geparsten Dokument. Die Callbackfunktion muss sich dann die benötigten Daten aus Strukturen wie der eben gezeigten erarbeiten. Im Falle eines Add Befehl wird die Zeichenkette, die ab Position content steht, in eine Datei geschrieben. Der Dateiname ist unter source->locURI gespeichert.

Umfangreicher ist die Funktion myHandleCopy(), weil mit ihr mehr als ein Datensatz bzw. eine Visitenkarte zum Client kopiert werden kann. Als Dateiname kann vom Client eine Zeichenkette wie "*.vcf" oder "herr_*_aus_*.vcf" angegeben werden. Es ist also möglich, das der Client die gesamte Datenbasis anfordert. Die Auswertung, welche Dateien wirklich zum Client gesendet werden sollen, überlässt der Server dem "ls" Befehl. Von ihm erhält er die genaue Liste, liest jede der Dateien und fügt sie einem dafür vorbereiteten SyncML Dokument hinzu. Es wird deshalb vor dem Parsen des empfangenen Dokumentes eine zweite Instanz im Workspace angelegt, um hier das Antwortdokument für den Client zu generieren.

Die Callback-Funktion myHandleStartSync() wird beim Parsen aufgerufen, enthält bei mir aber keine Funktionen. Hier könnte aber bei einer Adressbuchimplementation das Arbeitsverzeichnis für Add, Delete, und Copy festgelegt werden. Damit wäre es möglich, vielen Nutzern ein privates Adressbuch auf einem Server einzurichten.

6.4.4 Ausgabe

Eine Aufbereitung der übertragenen Dateien erfolgt für die Darstellung als HTML Datei. Der Inhalt wird hierzu nicht geparst, sondern nur mit einer Formatierung versehen, um die Darstellung lesbar zu machen. Es ist also möglich mit dieser Gästebuchrealisierung verschiedene Dateiformate zu übertragen, oder sich auf ein anderes als das vCard Format zu einigen. Einzige Restriktion ist, das es sich dabei um Texte handeln muss, denn die Dateien werden zeilenweise ausgewertet.

Gästebuch

Ausgabedatei der XML - SyncML Prototypimplementation zur Diplomarbeit
Synchronisation von Terminplanern mittels XML

```
klaus.vcf letzte Veränderung Okt 27 12:11

begin:vcard
n:Mustermann;Klaus
tel;cell:+49 (0) 177 991234
x-mozilla-html:FALSE
url:http://www.km.de
org:Mustermann Inc.
adr:;;Musterstr. 88;Musterstadt;;88888;
version:2.1
email;internet:Klaus.Mustermann@km.de
x-mozilla-cpt:;18912
fn:Klaus Mustermann
end:vcard
```

Ladevorgang abgeschlossen

Abbildung 6.5: Darstellung der vom Server erzeugten Ausgabe

Die gezeigte Ausgabe enthält eine vCard, deren Dateinamen und die Angabe, wann die Datei zum letzten Mal verändert wurde. Mögliche Ergänzungen, die je nach Einsatzszenario hinzugefügt werden könnten, sind Angaben über die Erreichbarkeit des Servers. Dieser könnte z.B. eine Meldung hinzufügen, wenn er gerade läuft und auf welchem Port er erreichbar ist. Auch Sortierungen nach anderen Gesichtspunkten als dem Dateinamen sind beim Einsatz der Anwendung anzupassen.

Kapitel 7

SyncML Produkte

7.1 Geräte

7.1.1 Nokia 9210

Das das Nokia 9210 kein gewöhnliches Mobiltelefon ist, fällt spätestens auf, wenn das normale Telefon zum Minicomputer aufgeklappt wird. Die Vorgänger dieses Communicator machten schon auf sich aufmerksam, als der Begriff “Smart Phone” noch gar nicht geprägt war. Vom 333 Seiten starken Benutzerhandbuch befassen sich gerade mal 30 Seiten mit dem Telefonieren, es muss also wichtigere Gründe geben, sich für dieses Gerät zu entscheiden.

Der Communicator enthält Textverarbeitung und Tabellenkalkulation. Mit ihm lassen sich Präsentationen erstellen. Faxe Mails und SMS können versendet und empfangen werden. WWW und WAP Zugang sind auch möglich. Bei einem Gerät mit diesen Fähigkeiten sind Adressbuch und Terminplaner natürlich auch auf entsprechend hohem Niveau angesiedelt.

Das Adressbuch kann mit Kontaktdaten, die auf der SIM Karte des Telefons, auf speziellen Speicherkarten und im Speicher des Communicator untergebracht sind, umgehen. Diese Kontaktdaten lassen sich bearbeiten und zwischen den verschiedenen Speicherbereichen kopieren. Auch das Exportieren als Visitenkarte im vCard Format ist möglich. Hierzu wird die Version 2.1 verwendet.

Eine angenehme Ergänzung des Terminkalender ist es, das Einträge in die ToDo-Liste mit einem Fälligkeitsdatum versehen werden können. Der Terminkalender verfügt über eine Reihe verschiedener Ansichten. Damit behält man auf dem verhältnismässig kleinen Display die Übersicht.

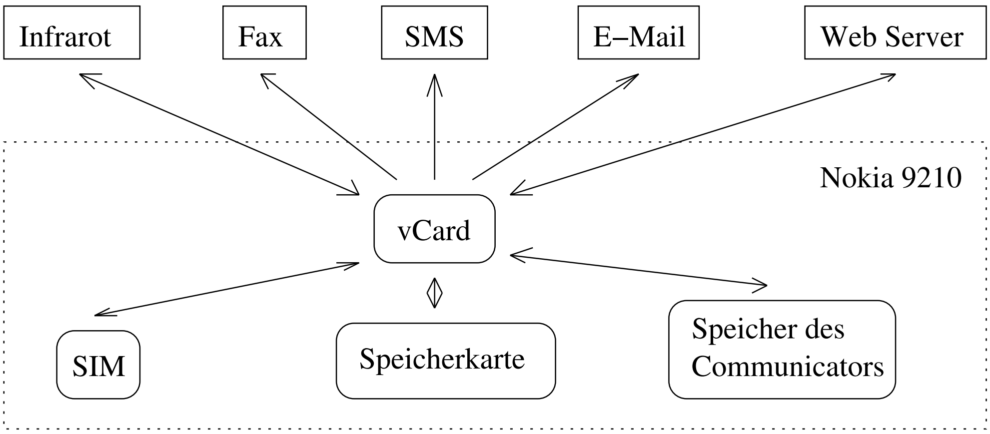


Abbildung 7.1: vCard Austausch mit dem Nokia 9210

Das der Nokia 9210 Communicator SyncML tauglich ist, steht nicht im Benutzerhandbuch. Für den Anwender stehen selbstverständlich mehr die Fähigkeiten als deren Realisierung im Vordergrund. Das Nokia 9210 wurde bereits im Dezember 2000 kurz nach Veröffentlichung der SyncML Protokolle als SyncML tauglicher Client angekündigt. Erhältlich ist das Gerät etwa seit der CeBIT im März. Die Zertifizierung als SyncML Client erfolgte jedoch erst im April auf dem ersten SyncFest. Synchronisiert werden können vCard 2.1 und vCalendar 1.0 . Als Transportprotokoll wird leider nur HTTP unterstützt. Für den lokalen Datenabgleich mit dem PC setzt Nokia auf eine eigene Software mit serieller- oder Infrarot-Verbindung.

7.1.2 Erricson T39m

Das Erricson T39m [30] ist ein Tri-Band Mobiltelefon und für schnellen Datentransfer über HSCSD und GPRS konzipiert. Als Kommunikationsschnittstelle zu Endgeräten ist Bluetooth vorgesehen. Der Personal Information Manager des Gerätes verfügt über Adressbuch und Terminkalender. Ein E-Mail Client ist auch vorhanden.

Im Terminkalender kann zwischen verschiedenen Ansichten gewechselt werden. Das Adressbuch verfügt mit 510 möglichen Kontakten über ausreichend Kapazität, um auch grössere Datenmengen aufnehmen zu können.

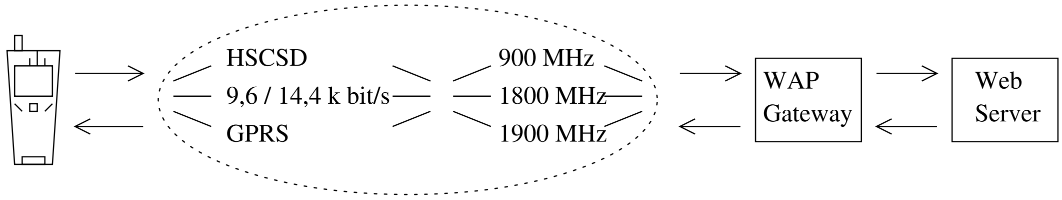


Abbildung 7.2: SyncML über WAP

Das T39m enthält einen SyncML Client. Damit ist es möglich, Kontaktadressen und

Kalenderdaten über WAP zu synchronisieren. Der Client wurde auf dem SyncFest im April 2001 zertifiziert und gehört damit zu den ersten markttauglichen Implementierungen.

7.2 Software

7.2.1 TrueSync Technology Platform

Die TrueSync Technology Platform ist ein Produkt der Starfish Software, Inc. . Mit diesem Produkt [25] wird versucht die komplette Datensynchronisation, welche in einem Unternehmen anfällt, zu übernehmen. Das Produkt besteht daher aus einer ganzen Reihe von Applikationen.

Der TrueSync Synchronization Server unterstützt die Datensynchronisation zwischen SyncML Servern, Clienten und anderen TrueSync Produkten. Die TrueSync Companion Software beherrscht die Datensynchronisation zwischen verbreiteten Windows PIM und den durch die Software unterstützten Geräten. Die TrueSync Companion Software enthält ausserdem selbst einen Organizer für den Fall, das der Kunde noch nicht über ein solches Programm verfügt.

Mit den TrueSync Device Solutions werden Geräte nachträglich SyncML konform. Das unterstützt auch die optimierte Synchronisation zwischen Starfish Clienten und Servern. Es ist also anzunehmen, das die Übertragung im WAP Binary Format erfolgt, und je nach Gerät durch den Einsatz von SyncML ein Vorteil gegenüber den proprietären Protokollen erreicht wird. Wer TrueSync Funktionalität in eigene Anwendungen integrieren möchte, kann das mit einem SDK tun und erhält damit die Möglichkeit, nicht nur Desktopanwendungen, sondern auch mobile Kleingeräte mit SyncML auszustatten. Als Geräte werden unter anderem Palm-OS, Windows CE und eine Reihe von Mobiltelefonen unterstützt. Da bei den Mobiltelefonen die Software nicht geändert werden kann, muss hier über das eingebaute Potokoll Synchronisiert werden. Bei diesen Geräten betrifft die Synchronisierung ausschliesslich Adressdaten.

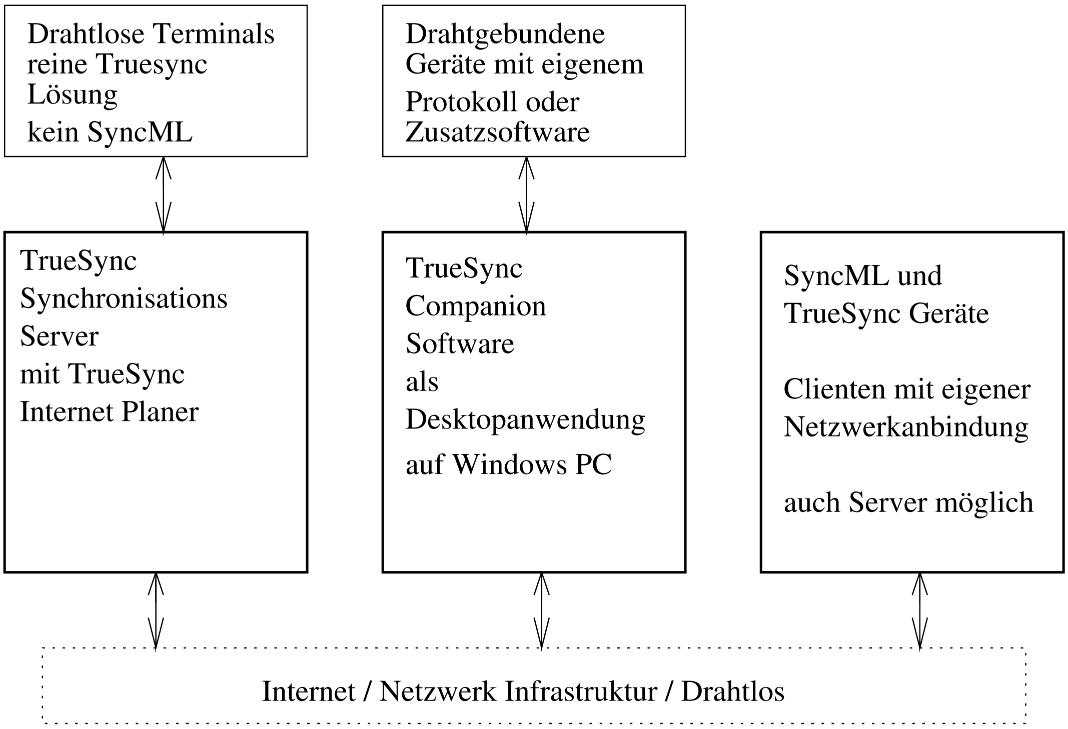


Abbildung 7.3: TrueSync Technology Platform

Bei TrueSync wurde also eine bestehende Synchronisationssoftware erweitert, um SyncML Funktionen bereitzustellen. Das hat den Vorteil, das der Teil den der Nutzer sieht langfristig unter ergonomischen Gesichtspunkten entwickelt werden konnte, um dann in einem entsprechend kürzeren Zeitrahmen SyncML zu integrieren. Wenn man von dem Ziel ein einheitliches Protokoll zur Datensynchronisation zu etablieren ausgeht, dann ist die Zahl der Parser und Filter, die hier integriert sind, noch sehr hoch. Dieser Overhead ist jetzt sicher noch nötig und die Zahl der integrierten Protokolle wird sich so verringern, wie verbundene Geräte selbst das einheitliche Protokoll unterstützen.

Kapitel 8

Ausblick

8.1 Wachstumsmöglichkeiten für XML basierte Synchronisation

SyncML legt nur fest, wie Daten übertragen werden. Das Format der Nutzdaten kann aber recht frei gewählt werden. Diese Datenformate ergeben sich aus dem Umfeld in dem SyncML eingesetzt werden soll. Weil die SyncML-Entwicklung von Mobiltelefon- und PDA-Herstellern vorangetrieben wird, und dort mit Adressbeständen, Terminen und ToDo-Listen gearbeitet wird, sind das die ersten Datenformate, die zum Einsatz kommen. Hier geht es um den Austausch von wenigen aber wichtigen Daten, die gleichzeitig auf unterschiedlichen Geräten bearbeitet werden sollen. Die verwendeten Geräte sind Massenprodukte, welche, auch wenn sie von unterschiedlichen Anbietern stammen, in ihrer Funktionalität ähnlich sind.

Die nächste Welle von Geräten die SyncML einsetzen, wird Geräte betreffen, die weniger der Informationsbearbeitung im Gerät dienen und mehr dem Sammeln von Daten. Die Information fliesst überwiegend vom Gerät zu einem Server, in umgekehrter Richtung wird die Verbindung zu Managementzwecken benutzt. Ein Beispiel für einen Vertreter dieser Gerätekategorie ist die Digitalkamera.

Heutige Digitalkameras werden über eine serielle Verbindung (RS232, USB, IrDA) an den PC angeschlossen, um Bilder oder Filme zu übertragen. Die Software wird vom Hersteller geliefert und funktioniert meist nur mit der mitgelieferten Hardware. Die Bedingungen für einen Datenabgleich sind also ähnlich derer von PDA´s mit Terminplaner- und Adressbuchfunktionalität. Digitalkameras könnten demnach von einer Standardisierung in diesem Bereich profitieren. Erste Ansätze für eine Zusammenarbeit zwischen Digitalkamera und SyncML sind in den Kombinationen Kodak PalmPix [19] mit 3Com Palm-Geräten und Nokia Komunikator 9210 [20] mit Digitalkamera, über Infrarot, zu finden.

Bei der PalmPix Kamera werden die Bilder im Speicher des Palm abgelegt. Wenn Palm-Geräte über SyncML verfügen, ist auf diesem Wege auch das Übertragen von Bildern an den PC möglich. Die Palm Inc. gehört der SyncML Initiative an. Es sollte deshalb nur eine Frage von Monaten sein, bis SyncML Funktionalität für den Palm zur Verfügung steht.

Im Nokia Communicator 9210 ist SyncML bereits implementiert. Die Kommunikation zur Digitalkamera erfolgt über die Infrarotschnittstelle. Auch beim Communicator ist, wie beim Palm, eine extra Software für die Kommunikation zur Kamera und Management der Bilder nötig. Ob die Bilder bei der Synchronisation mit SyncML berücksichtigt werden, hängt von den verwendeten Datentypen (jpeg, bmp, ...) ab. Diese Datentypen sind zu den Gerätefähigkeiten hinzuzufügen, die beim Aushandeln der SyncML Verbindung übertragen werden. Der Client muss dem Server mitteilen, das er mit den Bilddaten umgehen kann, wenn er auch Bilddaten empfangen möchte. Der Server wird ohnehin alle Daten des Clienten entgegennehmen. Der Einwegcharakter der Datenübertragung zum Server wird durch den SyncML Befehl "Archive" unterstützt. Die Daten werden damit vom Clienten auf den Server verschoben, um sie dort zu sichern, und im Clienten Platz zu schaffen.

Die Verwendung eines PDA ähnlichen Gerätes als Vermittler zwischen Kamera und Server ist nötig, weil sich bei heute verfügbaren mobilen Digitalkameras kein Softwareupdate durchführen lässt. Deshalb ist es auch nicht möglich, die Kamera um SyncML Funktionen zu erweitern. Sollte das einmal der Fall sein, dann sind PDA´s oder Mobiltelefone in diesem Zusammenhang nur noch als Gateway nötig, um den Transport der Daten sicherzustellen.

Ein Szenario für diese Gerätekombination könnte im journalistischen Umfeld die Anfrage des Redaktionsservers an die Kamera eines Fotografen sein, um somit die Synchronisation der Bilddaten einzuleiten. Der Fotograf arbeitet unter Umständen noch und muss sich nicht um die Bildübermittlung kümmern.

Was ich eben am Beispiel der Digitalkamera gezeigt habe, gilt für viele Bereiche in denen Daten anfallen und mit proprietären Mechanismen übertragen werden. Weitere Einsatzgebiete, für einen XML basierten Datentransfer, könnten in der Datenerfassung, z.B. zur Sicherung technischer Anlagen, in Alarmanlagen, Wetterstationen und so weiter liegen. An dieser Stelle handelt es sich bei den Clienten jedoch nicht mehr um PIM, sondern um "unbemannte" Technik. Den Vorteil für die Datenübertragung sehe ich darin, das ein Server eine Vielzahl unterschiedlicher Clienten bedienen kann, weil auf proprietäre Protokolle verzichtet werden kann. Die Nutzdaten könnten dann in einer weiteren XML Sprache beschrieben sein und unter Umständen gleich mit vom Kommunikationsserver ausgewertet werden.

Die Möglichkeit ein Synchronisationsprotokoll über verschiedene Transportprotokolle zu verwenden, kann zur Schaffung von Redundanz bei der Kommunikation verwendet werden. Das kann entscheidend dafür sein, ob die Kommunikation zustande kommt oder, zu welchem Preis eine Synchronisation erfolgen kann. Eine Verbindung des Notebooks über das Mobiltelefon kann fast überall hergestellt werden. Zu Hause ist aber vielleicht die Verbindung über Bluetooth und DSL günstiger.

8.2 Künftige Anforderungen an einen Terminplaner

Ich möchte auf Anforderungen an einen Terminplaner eingehen und nicht auf die Anforderungen an einen PIM. Die Entwicklung der PIM ist (noch) zu stark von den technischen

Möglichkeiten der Produktion abhängig.

Computerentwicklung wird vorrangig in der westlichen Welt betrieben. Hier ist der gregorianische Kalender gültig. Als Folge davon arbeiten Terminplaner auf der Basis dieses Kalenders. Wer ein Programm zur Umrechnung zwischen verschiedenen Kalendern sucht, findet in [32] ein Programm, das die bekanntesten unterstützt. Verschiedene Kalender einzusetzen, ist in der Regel nicht üblich. Der gregorianische Kalender ist bezüglich Genauigkeit und Einfachheit kaum zu verbessern. Weil aber z.B. die 7-Tage Woche nicht in das Monats- und Jahresschema passt, gibt es seit Mitte des letzten Jahrhunderts verschiedene Vorschläge zur Einführung eines neuen Weltkalenders. Die Probleme bei der Einführung eines solchen würden das Jahr 2000 Problem in den Schatten stellen (vielleicht hätte es dann auch nie existiert). Also selbst, wenn sich an der maschineninternen Darstellung von Datumsangaben nie wieder etwas ändern sollte, könnte durch eine stärkere Trennung zwischen interner Darstellung und der externen Repräsentation mehr Flexibilität erreicht werden. Der buddhistische Kalender ist uns nämlich um 543 Jahre voraus, während wir laut islamischem Kalender das Jahr 1422 haben. In einigen hundert Jahren ist vielleicht auch der Terminkalenderabgleich nach dem Marskalender nötig. Doch das ist selbst in einem Ausblick Zukunftsmusik.

Ein wichtiges Kriterium für den Einsatz eines Terminplaners wird künftig dessen Teamfähigkeit sein. Aus heutiger Sicht ist es wichtig, das der Anwender mit seinen eigenen Daten umgehen kann, damit meine ich die technische Handhabung der Informationen. Bei der Lösung dieses Problems kann SyncML eine wesentliche Rolle spielen. Die Teamkoordination wird dabei der Sync-Engine eines Server überlassen. Wenn es sich aber um ein gerade entstandenes zwei Personen Team handelt, wäre vorstellbar, das der PIM des einen die Serverfunktionalität übernimmt. Auch ein Aushandeln eines Termines bei dem jeder Kommunikationspartner so wenig wie möglich Informationen preisgibt, ist vorstellbar. Nach der Realisierung der Teamfähigkeit könnte ein dezentralisierte Synchronisation der nächste Schritt sein.

Abkürzungsverzeichnis

API	Application Program Interface
DOM	Document Object Model
DTD	Document Type Definition
GPRS	General Packet Radio Services
GUID	Global Unique IDentifier
HSCSD	High Speed Circuit Switched Data
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IMEI	International mobile station equipment identity (die Fahrgestellnummer des Handys)
IrDA	Infrared Data Association
J2ME	Java 2 Micro Edition
LUID	Local Unique IDentifier
OBEX	OBject EXchange protocol
PDA	Persönlicher Digitaler Assistent
PIM	Personal Information Manager
RTK	Reference ToolKit
SAX	Simple API for XML
SCTS	SyncML Conformance Test Suite
SIC	SyncML Interoperability Committee
SIM	Subscriber Identification Module
TCP	Transport Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WBXML	WAP Binary XML (Wireless Application Protocol Binary Extensible Markup Language)
WML	Wireless Markup Language
WSP	Wireless Session Protocol
XML	Extensible Markup Language

Literaturverzeichnis

[2] http://www.imslsoft.com/xlinkwin.htm
connectivity products for Casio and Sharp electronic organizers
Autor: IMSL Software

[3] http://www.mus.ch/falter/2000-04/PDA.html
Die Geschichte des Palm
Autor: Susie Seiler
Datum: 04/2000

[4] http://www.apple-history.com/german.html
Geschichte
Autor: Glen Sanford
Datum: 01/2001

[5] http://grillennetz.at/suppen2/witze1.htm
Witze 1 - Süppenhühns Web # 2

[6] http://www.pstec.de/ppp/ppppowpi/ppppowpi.html
PPP - Power for the Palm
Autor: Peter Strobel
Datum: 03/2000

[7] http://mobilix.org/pda_linux_palm.html
Linux Applications for the Palm and Visor PDA
Autor: Werner Heuser
Datum: 10/2001

[8] http://www.nokia.de/service/mobile_phones/faqs/kalender.html
Nokia - Kalender in Nokia Mobiltelefonen
Datum: 2001

[9] http://www.gnome.org
GNOME - Computing made easy
Datum: 10/2001

[10] http://www.gnu.org
GNU's Not Unix! - the GNU Project and the Free Software Foundation (FSF)

Autor: rluddy
Datum: 10/2001

[11] http://www.linux-magazin.de/ausgabe/1999/02/Gnome/gnome.html
GNU Network Object Model Environment - Auf den Spuren des Yeti
Autor: Markus Schmitt
Datum: 02/1999

[12] http://www.syncml.org/list-supporters.html
List of supporters
Autor: SyncML Initiative Ltd.
Datum: 2001

[13] http://www.syncml.org/docs/syncml_represent_v101_20010530.pdf
SyncML Representation Protocol
Autor: SyncML Initiative Ltd.
Datum: 05/2001

[14] http://www.syncml.org/docs/syncml_protocol_v10_20001207.pdf
SyncML Sync Protocol
Autor: SyncML Initiative Ltd.
Datum: 12/2000

[15] http://www.awb.at/e_abc.html
e-Definition, Das ABC des e-Commerce
Autor: Austrian Web Business
Datum: 2000

[16] http://www.palm.com/products/accessories/enterprise/server.html
Palm HotSync Server
Autor: Palm Inc.
Datum: 2001

[17] http://www.pumatech.com/Intellisync_Palm_CE.html
Pumatech: Intellisync
Autor: Pumatech, Inc.
Datum: 2001

[18] http://www.synchrologic.com/about/about_imobile.html
iMobile Suite: Total Mobile and Wireless Infrastructure
Autor: Synchrologic, Inc.
Datum: 2001

[19] http://www.kodak.com/US/en/digital/cameras/palmPix/
Kodak: PALMPIX Gateway

Autor: Eastman Kodak Company
Datum: 2001

[20] http://www.nokia.com/phones/9210/
Nokia 9210 Communicator
Autor: Nokia
Datum: 2001

[21] http://www.syncml.org/downloads.html
Downloads
Autor: SyncML Initiative Ltd.
Datum: 2001

[22] http://groups.yahoo.com/group/SyncML/message/58
Re: SyncML examples
Autor: Peter Thompson
Datum: 08/2001

[23] http://www.smartner.com/products/engine_datasheet.html
smartner - products - technology modules - smartner engine
Autor: Smartner Informations Systems Ltd.
Datum: 2001

[24] http://mordor.cs.hut.fi/tik-76.115/00-01/palautukset/groups/SyncML
/t3/projplan/project_plan.html
SyncML Project Plan
Autor: Antti Saarilahti
Datum: 02/2001

[25] http://www.starfish.com/solutions/ts_tp.html
Starfish Software: TrueSync
Autor: Starfish Software, Inc.
Datum: 2001

[26] http://sync4j.sourceforge.net/
sync4j homepage
Autor: Sean Sullivan
Datum: 2001

[27] http://sourceforge.net/projects/libsyncml/
SourceForge: Project Info - LibSyncML
Autor: Max Berger
Datum: 2001

[28] http://sourceforge.net/projects/syncmlsdk/
SourceForge: Project Info - SyncML SDK
Datum: 2001

[29] http://gartner11.gartnerweb.com/public/static/hotc/00089713.html
Can SyncML Stay in Sync?
Autor: K. Knox
Datum: 07/2001

[30] http://www.ericsson.at/presse/2001/pi_jul06.html
Facts: Ericsson T39m
Autor: Ericsson Österreich
Datum: 07/2001

[31] http://www.linux-magazin.de/ausgabe/1997/07/Plan/plan.html
Kalender und Terminplaner - Planmäßig
Autor: Thomas Driemeyer
Datum: Juli 1997

[32] http://www.computus.de/kalenderprogramm/kalenderprogramm.htm
Die Kalenderprogramme Unikal und Skal
Autor: Herbert Metz
Datum: 10/2001

Selbständigkeitserklärung

Ich erkläre, das ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Chemnitz, den 30. Oktober 2001