



**Swiss Mac Meeting 2005, 5.11.2005, Basel: Notizen & Handout zum Vortrag**

**Vortrag 2: Software-Entwicklung in der Schweiz (mit WebObjects): Erfahrungen, Chancen & Risiken - Marc Oesch, Sendai Systems**

**1. Warnung...oder was dieser Vortrag nicht sein soll**

- Software-Entwicklung auf Mac OS X (Cocoa) betrachten oder Software-Auftragsarbeit (=rein kundenspezifische Software als Auftrag).

Cocoa-Interessierte müssen nicht gleich den Raum verlassen, denn es soll auch nicht eine...

- Verkaufsveranstaltung für unsere Software, die auf WebObjects basiert,

- Technische Einführung in WebObjects (kann bei Interesse später nachgeholt werden mit anderem Referenten)

...werden.

Zudem ist dieser Vortrag weder von Apple autorisiert noch empfohlen, er soll bewusst auch negative und kritische Punkte aufzeigen.

**2. Dennoch kurz: Was ist Apple WebObjects ? Was entwickeln wir (Sendai Systems, Bern) mit WebObjects ?**

Was ist WebObjects ?

"Middleware"-Software: Von NeXT übernommen zusammen mit Openstep und Steve Jobs. Apple "kaufte" also gleich drei gute Dinge.

Heute Applikationsserver von Apple, meist mit SQL-Datenbanken als Quelle: Erstellen von Webapplikationen oder netzwerkverteilten Applikationen im weiteren Sinn. WebObjects-Applikation hat in der Praxis häufig ein HTML-Frontend, dies muss aber nicht immer der Fall sein (Generieren von XML/Webservices, Anbinden an native Applikationen wie Apple iTunes, Java Desktop Clients...).

WebObjects ist auch Apples "bestgehütetes Geheimnis" (Zitat News.com) <sup>1</sup>. Dies wird sicher auch so bleiben, WebObjects ist seit der WWDC 2005 Teil der Entwicklungsumgebung Xcode / Developer Tools. WebObjects ist also kein eigenes Produkt mehr, mit dem Apple Geld verdienen will; wird wohl in Zukunft noch enger an Apple-Technologie verknüpft, trotz Java als Programmiersprache. Apple will damit wohl (Server-)Hardware verkaufen, daneben ist WebObjects Grundlage der eigenen Apple Web-Anwendungen. Dienste wie DotMac, der Apple Store und der iTunes-Store basieren alle auf WebObjects...**fast alle Apple-User nutzen also WebObjects passiv.**

Was macht Sendai Systems mit WebObjects ? (Server-)Softwarepaket GroupVille. Funktionalität von Groupware wie Gruppen-Kalender, Webmail, Wiki, Kontaktverwaltung, Aufgaben, Notizen. (Siehe unter 7. für mehr Informationen zu Firma Sendai und der Software). GroupVille 2.x ist eine "klassische" WebObjects-Applikation: Web-basiert mit HTML-Frontend und etwas Javascript.

**3. Zweite Warnung: Macht es nicht...oder wer will 70 Stunden pro Woche arbeiten und bleich wie der Vortragende aussehen :) ?**

Die Dotcom-Zeit ist vorbei...

- Firma gründen, Risikokapital ohne auch nur im Ansatz fertiges Produkt bekommen
- Werbeagentur suchen, Produkte gross ankündigen
- Börsengang planen und durchführen
- The Future is Fantastic ! (Fantastic als Paradebeispiel eines Schweizer DotCom)

...und das ist gut so.

Diese Punkte wurden damals völlig vernachlässigt:

- Ist ein Markt vorhanden ? - Fokus auf gutes Produkt und die Produktentwicklung ?
- Entwickelt und immer wieder getestet gemäss Kundenbedürfnissen ?

Wer heute eine (Software-)Firma in der Schweiz gründet oder betreibt, macht es meist auch aus Idealismus. Man muss gewillt sein, einige Monate/Jahre sehr viel zu arbeiten und wenig zu verdienen.

Bei reinen Auftragsarbeiten für Kunden ("Body leasing"), die im Stundentakt nach Aufwand verrechnet werden, ist die Situation wohl besser. Hier müssen aber ständig neue Kunden gefunden werden, wenige Polster in schlechten Zeiten...

Folgende Punkte sind meiner Meinung nach wichtig bei Interesse Software in der Schweiz zu entwickeln :

- Eigene Firmen bedeutet nicht nur Software entwickeln (Administration, Finanzen, Kundenbeschwerden...). Will ich wirklich selber eine Firma gründen ? Bin ich die richtige Person dazu ? Gute Links, Tests und Unterlagen dazu unter <http://www.ifj.ch/> (Institut für Jungunternehmer).

Meine Erfahrung: Team aus mindestens 2 Personen ist oft besser als eine, zb. Aufteilung in Programmierung/Design und Verkauf/Support.

- Kapital ist in der Schweiz viel vorhanden...aber **nicht** für Jungunternehmer, Die Lage ist zwar Ende 2005 wieder etwas besser als 2001-2003. Dennoch sehr wenig Fremd- oder Eigenkapital, wenn kein "Ausweis" vorhanden. Mögliche "Ausweise" sind:

- Person schon älter (40-50 Jahre), Kapital und/oder Kontakte vorhanden aus seiner früherer Tätigkeit, die Start ermöglichen und gleich erste Kunden sind.

- Bereits eine Firma gegründet, oder volles Team von 5-10 Leuten vorhanden.

Software-Entwickler, die ich in der Schweiz kenne, haben diese Voraussetzungen meist nicht, sind "Einzelkämpfer oder kleinere Teams.

Vor allem Zeit bis zur ersten Version 1.0 muss selbst finanziert werden. Ideen in diesem Falle :

- Erste Kunden aus Praktikum oder aus Uni/Fachhochschule mitnehmen (zB. über Diplomarbeit).
- Teilselbständigkeit (zum Beispiel 50% weiter auf letzter Stelle arbeiten) bis Version 1.0, Freunde/Bekannte/Verwandte.

- Sobald man genug Umsatz erreicht nach Version 1.x, muss oft eine Person für den Support eingestellt werden. Software ist meist supportintensiv, kein Selbstläufer. Siehe dazu <sup>2</sup>

Alle obigen Restriktionen haben aber auch Vorteile: Keine unnötigen "Gags" mehr bei der Entwicklung oder im Marketing. Fokus auf Kundenwünsche und Kundennutzen in der Software. Auch scheint es nach dem Börsengang von Google neue Aufmerksamkeit zu geben, Schlagwort Web 2.0 ("Social Software" wie Blogs und Wikis, Web-Technologien wie AJAX / XUL / XAML ..)

#### 4. Die Software ist fertig...oder ich gehöre zu den glücklichen 30%. Wie komme ich nun auf den Markt und an die Kunden ?

Einige wichtige Punkte als Stichworte (nicht vollzählig - dazu fehlt leider die Zeit), die studiert werden sollten:

4.1 **Ertragsmechanik...oder wie verdiene ich das liebe Geld ?** Will ich eine **Produktfirma** (eher Microsoft als prominentes Beispiel) oder eine **Servicefirma** (eher IBM als prominentes Beispiel) sein ? Soll das Produkt closed source oder open source sein ?

Produktfirma: Umsatz durch Lizenzen. Support durch Dritte, Händlernetz muss aufgebaut werden - oder Software braucht wenig Support (Beispiel Game-Software).

Servicefirma: Fokus auf lokaler Schulung/Support oder rein kundenspezifischer Software nach Auftrag (Beispiel "eigener" oder fremder Open Source, hier oft fast 100% Umsatz durch Support/Schulung oder Auftragsarbeiten). Dies kann aufgrund geographischer Umstände meist nur in der Schweiz geschehen, sonst Reisespesen etc. für den Kunden zu hoch.

Schätzung heute: 70-75% Umsatz mit Services und Support, 25-30% mit Lizenzen; dies wird sich in Zukunft weiter verschieben in Richtung Services aufgrund des steigenden Einsatzes von Open-Source Software.

4.2 **Zeit bis Release 1.0 nutzen** : Etwa 70% aller Softwareprojekte scheitern, das heisst kommen erst gar nicht auf den Markt...

Sich keine Illusionen, dass man diese Statistik ganz schlagen kann. Einige Vorsichtsmassnahmen:

- Alle erstellen Zeitpläne mit Faktor 3 multiplizieren...mindestens :)
- Kleiner Release 1.0, der ausbaubar ist, ist wichtig: "Writing code isn't hard, maintaining code is hard !"
- Dokumentation früh planen und auch Zeit, um Dokumentation zu erstellen: Nach Release noch weniger Zeit dafür !
- Früh Betas verteilen, vorerst nur an geschlossene Benutzergruppen. Qualität wird durch Endbenutzer definiert, Feedback ist wichtig...Entwickler sieht oft den Wald vor lauter Bäumen nicht mehr., verschwendet Zeit an nicht relevanten Features oder Bugs.  
Weiterer Vorteil: Potentielle Kunden kennen das Produkt bereits, wichtig, da oft lange Entscheidungs- / Evaluationen.
- Nach Version 1.0 wird alles noch schlimmer: Abhängigkeiten steigen bei neuen Versionen (siehe Microsoft und Altlasten bis zurück zu MS-DOS als Extrembeispiel..)

4.3 **Zielgruppen** für mein Produkt definieren: Individuelle, Klein- und Mittelbetriebe oder Grossfirmen ? Massenmarkt oder Nische ?

- Eine grosse Firma kann Marktstudien und -forschung etc. durchführen. Als kleine Firma schon nur aus Kostengründen nicht möglich, deshalb: Online (Newsforen) und Magazine, persönliche Bekannte fragen oder beoachten nach Bedürfnissen / Wünschen, die nicht gedeckt sind.

- Konzentration auf einige wenige Kunden ist besser, Eintritt und Bestehen in den Massenmarkt (siehe Abb1 unten) ist nur mit viel Kapitaleinsatz möglich, das heute nur schwierig beschaffen werden kann. Beispiel: Wieviele grosse Auktionsseiten gibt es noch ?

- Bei (grösseren) Firmenkunden sollte man die internen Abläufe kennen:

- Entscheidungsträger und Budget: Wer fällt am Schluss den Entscheid für den Kauf ? Wer beeinflusst den Entscheider ?
- Dauer der Entscheidungsfindung insbesondere Grossfirmen kann **sehr** mühsam sein, keine Ressourcen, als Kleinfirma, den Kontakt über Monate zu pflegen bis zum Abschluss. Vorsicht vor klingenden potentiellen Kundennamen !

- Das Internet ist der beste Marketing-Freund der kleinen Firma geworden:

- Erfahrungsaustausch: Direkter Kundenkontakt und Feedback durch Interessenten / Kunden.
- Werbung (etwa in Google Ad Words): 1: 1 Marketing möglich für Kleinstfirmen durch Schlagworte !
- Online-Abrechnungssysteme, Online-Druckereien etc.: Auslagern von nicht zentralen Elementen einfach.

4.4 **Langfristigkeit und Ausbaubarkeit**: Ich habe einige Lizenzen verkauft. Was mache ich nun (siehe dazu auch oben Punkt 4.1) ?

Gute Produkte finden immer einige erste Abnehmer, aber wie komme ich nun weiter ? Folgende Probleme:

- (Grosse) Firmen sind träge...oder: Kennen Sie noch "Alphablox" ?

Einschub: Alphablox entwickelte "Rich Internet Applications" schon vor etwa 5 Jahren (heute wieder in Mode unter dem Stichwort AJAX und dank Anwendungen wie Gmail oder Google Maps von Google). Alphablox war so als kleine Firma 5 Jahre zu früh auf dem Markt !

- (Grosse) Firmen sind risikoscheu (siehe dazu auch Punkt 5, spricht gegen Einsatz von Apple-Technologie)

Zusammengefasst: Wie komme ich bis zum Aufstieg oder über den Berg bzw. überwinde die Lücke (*Chasm*) zwischen den Frühkäufern (Innovators und Early Adopters), die als kleine Minderheit gerne neue Produkte kaufen hin zur konservativen Mehrheit / Massenmarkt ?

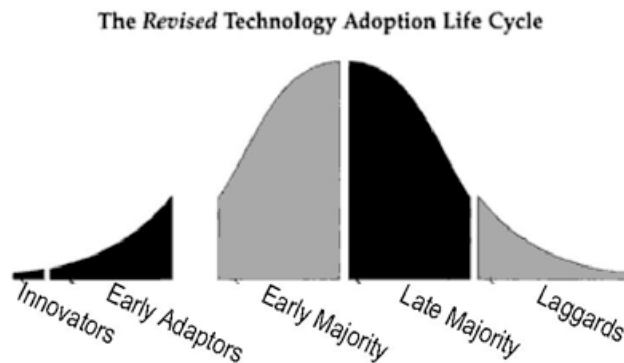


Abb 1: "Crossing the Chasm": Kauf neuer Technologien: Markteinführung nach Kundengruppen von links nach rechts.

Geduld oder viel Kapital ist nötig, um diese Lücke zu überspringen: Die ersten zwei Gruppen informieren sich auch selbst über Produkte und tauschen sich aus, fast kein Marketingaufwand notwendig. Dies ändert sich bei und nach der Lücke auf dem Weg zur "Early Majority".

- Als kleine Firma kann man eventuell auch alleine mit den "Innovators" und "early adaptors" überleben. Das obige Modell stammt aus den USA und richtete sich eher an schnell wachsende Firmen, die grössere Beträge über Investoren zur Verfügung hatten und sich an den Massenmarkt zu richten versuchen..

- Faustregeln aus eigener Erfahrung :

- Zuerst zufriedene kleine Kunden gewinnen, die eher Neues ausprobieren und schneller Kauf-)entscheide fällen, bevor man (konservativere) Grosskunden sucht. Grosskunden am Anfang / Referenzkunden sind natürlich ein Glücksfall...
- Wartungsverträge / Support / Schulungen für erste Kunden sind wichtige "Versicherung" nach Verkauf der Software-Lizenz. Generieren Umsatz, damit man das Produkt weiterentwickeln kann. Siehe Punkt 4.1 mit der "75% / 25%" Regel.

## 5. Apple / WebObjects spezifische Chancen & Risiken

- Kommunikationspolitik von Apple ist ein Nachteil. Auch mit unterschriebenem NDA (Non Disclosure Agreement) weiss man - als kleinerer Entwickler - oft nicht mehr als auf macrumors.com oder appleinsider.com steht. Beispiel 2005 : Wechsel zu Intel-Prozessoren. Beide Sites deshalb bookmarken, Pflichtlektüre...Macprime.ch natürlich auch, aber das wissen alle Anwesenden schon :)

- Emotionale Marke Apple als **Vorteil bei Privatkunden** / bestehenden Apple-Kunden:

- Sind meist sehr zufrieden mit Apple, interessiert an innovativer Software und hoher Qualität / Design.
- Weniger "Herdenverhalten" (böse ausgedrückt...) als unter Windows-Käufern üblich: Wer sich schon bewusst für eine Nischenplattform entschieden hat, wird auch eher kleinen Software-Firmen auf dieser Plattform eine Chance geben.

Nachteil: Software für Privatkunden darf nicht sehr viel kosten, meist ca. 50-100 Franken als Schmerzgrenze. Man muss sehr viele Lizenzen verkaufen, um damit über die Runden zu kommen. Zudem wollen Privatkunden meist nur eine Lizenz kaufen, nicht Geld für Support ausgeben (siehe Punkt 4.1).

Bei genügend Kunden muss dann neben Programmieren auch Support-Personen haben. Siehe dazu einen sehr interessanten Artikel im Blog in Daring Fireball <sup>2</sup>

- Emotionale Marke Apple als **Nachteil bei Firmenkunden**, Man muss nicht nur sein Produkt, sondern auch noch den Entscheid Apple "verkaufen". Am wichtigsten deshalb folgender Punkt in allen Marketing-Anstrengungen besonders beachten:

**- Produkt und Produktnutzen für den Kunden verkaufen, nicht die Technologie ! Der Entwickler ist oft zu technikverliebt. Der Kunde sollte gar nicht "unter die Haube" schauen müssen, der interessiert sich für den Nutzen bzw. sein Anforderungsprofil an die Software mit einer Muss- und Soll-Liste (was muss und soll die Software können): Diese Liste beachten statt eigener Featureliste.**

- Falls sich die Apple-kritischen Fragen doch nicht vermeiden lassen, folgende mögliche Antworten gegen Einwände zu Apple / OS X :

- "Apple ist teuer": Total Costs of Ownership (TCO) durchrechnen, nicht Hardwarekosten und erste Schulung oder "Würden Sie einen Lada als Lieferwagen kaufen, nur weil er in der Anschaffung billiger ist ?"  
Die meisten Firmen benutzen trotzdem Toyota oder Mercedes als Transporter; dieser korrekte Kostenvergleich nach den gesamten Lebenskosten und auch Nutzen (Return on Investment, ROI) - auch mit Vergleich zu Nullentscheid (weiter wie bis heute) - hat es in der Informatik immer noch schwer. Investitionsrechnung mit Apple Hard- und eigene Software lohnt sich bei Offerten. Apple ist in diesen Gesamtrechnungen meist nicht mehr teurer, bei Einbezug aktueller Softwareprobleme unter Windows (Viren, Adware...) sogar trotz Schulung von Windows-Usern oft günstiger.
- "Apple ist proprietär": "Ihre (grosse) Firma benutzt doch auch Sun Server und Solaris ?" Auch proprietäre Technologien. Proprietär hat bei der Abstimmung von Hard- und Software auch Vorteile bei der Wartung und Stabilität.
- "Apple macht zwar nette Spielzeuge wie iPods" / "Personen mit Apple-Erinnerungen an die Zeit vor OS X (Systemabsturz etc.)"

"Mac OS X ist gleich wie viele Gross-Systeme UNIX-basiert und sehr stabil. Sie haben doch auf Linux-Server ?".

- "Ist zwar gut, aber kennen wir nicht...". Schulung/Einführung/Integration in bestehende Infrastruktur anbieten: Windows-zentrierte Administratoren oder IT- Leute haben oft Angst um Ihren Einfluss, wenn nicht die Ihnen bekannte Windows-Technologie zum Einsatz kommen soll.

- Obige kritische Fragen sind ein Vorteil von WebObjects. Im Gegensatz zu nativer OS X-Software können beim Kunden weiterhin Windows-Clients mit Web-Browsern eingesetzt werden. Anstelle von Wechsel aller Clients nur ein (zusätzlicher) Server...

- Nachteil von WebObjects: Relativ unbekannt, es sind viel mehr Entwickler in PHP, .Net vorhanden. Einige Erweiterungen zu Java (WebObjects Frameworks müssen erlernt werden, bis am produktiv ist). Dafür nach Erlernen dieser Grenze sehr produktiv.

- Apple kann kleinen Entwicklern nicht helfen. Apple listet 20'000 Softwareprodukte, konzentriert sich auf wichtige / grosse Entwickler. Deshalb persönliche Kontakte (an WWDC, Mac World, andere Entwickler im gleichen Land...) aufbauen.

- "Geniale" Neuentwicklung auf OS X kann auch eine Gefahr sein (siehe Watson, Konfabulator). Apple entwickelt mehr und mehr Software selbst, wird zum Konkurrenten. Aktuelles positives Gegenbeispiel : Delicious Library (<http://www.delicious-monster.com/>)

- Apple/Macintosh als prozentual kleine Plattform am Gesamtmarkt. Wenig Marktanteil, aber folgende Vorteile zu anderen Plattformen:

- Windows: 20x Marktgrösse, aber auch 20-25x mehr Konkurrenz (andere Entwickler). Microsoft bedrängt eigene Entwickler noch mehr als Apple mit eigenen Produkten (Dominanz von Produkten wie Explorer, Office, Outlook). Leute haben unter Windows oft Angst, unbekannte Software zu installieren...oft komplett nur Microsoft-Produkte im Standard-Bereich und dazu selbst entwickelte Software.

- Linux: "Wir machen lieber alles selber". Oft Misstrauen gegenüber proprietärer Software / Lizenzkauf.

## 6. Schweiz-spezifische Chancen & Risiken

- Relativ hoher Marktanteil von Apple in der Schweiz (zusammen mit den USA, Niederlande und Japan).

- Schweiz ist zu klein, um im Produktverkauf von Software erfolgreich zu sein, man muss sofort international anbieten (prominente Ausnahmen gibt es wie immer : Abacus Research) oder sich auf auf Service/Support konzentrieren.

- Schweiz ist mehrsprachig und durchmischt : Anpassung an grosse und kleine Eigenheiten und Details sind für uns natürlich. Software-Entwickler aus anderen Ländern vernachlässigen dies oft. Beispiel: Warum muss ich bei Webseiten aus den USA nach Eingabe des Landes Schweiz eine "State/Province" zwingend eingeben ?

- Schweizer haben im internationalen Vergleich recht hohe Erwartungen an Qualität und Support: Preis ist in der Schweiz (anders als etwa in Deutschland) nicht das wichtigste Argument.

## 7. Interesse geweckt ? Fragen ? Kontaktinfos ?

- Technische Informationen zu WebObjects von Apple: Siehe Quellen unter <sup>3</sup>

- Informationen zum Vortragenden: Marc Oesch, Sendai Systems, Bern, [marc@sendaisystems.com](mailto:marc@sendaisystems.com)

- Informationen zu unserer Software GroupVille: <http://www.groupville.com> [Gratisversion für bis zu 3 User unter OS X Server]

## 8. Quellen / Abbildung

1 Suche in News.com: "WebObjects: Apple's best-kept secret ?", 1999: Siehe [http://news.com.com/WebObjects+Apples+best-kept+secret/2100-1001\\_3-228599.html](http://news.com.com/WebObjects+Apples+best-kept+secret/2100-1001_3-228599.html)

2 Daring Fireball "The Life": [http://daringfireball.net/2005/10/the\\_life](http://daringfireball.net/2005/10/the_life). Auszug:

*The full story is that like most forms of popularity, software sales tend to follow a power-law distribution. Most apps languish in obscurity, but popular apps tend to become super-popular. The hardest part about selling software is just getting noticed; if your app is only known by the sort of people who check VersionTracker and MacUpdate a few times a day, or even just the sort of people who know what VersionTracker and MacUpdate are, it's unlikely that you'll sell enough software to earn a living.*

*But once you get over that initial hump, subsequent sales become much easier. You start getting media coverage, and most importantly, assuming most of your users are happy, word-of-mouth advertising really starts to bloom.*

*But selling software isn't like selling books. When a book takes off and climbs the best-seller charts, that's just money in the author's pocket. Each software sale, on the other hand, comes with incremental support costs.*

*Users need help. They need help when they encounter bugs. [...]*

*So the conundrum is this: once a developer gets enough paying users to consider quitting his day job so he can devote full-time effort to writing code, he's quite possibly got so many paying users that he'll spend much of his time helping customers in ways other than writing code.*

3 <http://www.wocode.com>. Listet sowohl Code als auch alle relevanten Bücher für Entwickler zu WebObjects auf, gute technische Einstiegsseite.

Bei Apple unter <http://www.apple.com/webobjects/> [allgemeine Infos] sowie <http://developer.apple.com/tools/> [Code, Tools, Dokumentation]

WebObjects ist seit Release 5.3 unter den Entwickler-Tools aufgeführt und gratis als Teil von Xcode 2.x erhältlich.

Abb 1 Moore, Geoffrey: Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers, 1999