Platform Independent Mobile Application Development

Für verteilte und mobile Applikationen

Andreas Vogler
ETM professional control GmbH
A Siemens Company
Eisenstadt
andreas.vogler@me.com

Florian Schirxl
Schirxl Franz Sicherheitstechnik
Stockerau
Schirxl@msn.com

I. ABSTRAKT

Dieses Dokument beschäftigt sich mit der plattformunabhängigen Applikationsentwicklung. Sind die Ziele einer plattformunabhängigen Entwicklung von verteilten Applikationen auch im mobilen Bereich möglich.
Ausgang dieser Untersuchung ist die Schaffung einer globalen verteilten mobilen SCADA Applikation.
Aufgrund des enorm wachsenden mobilen Marktes ist die plattformunabhängige Entwicklung von Applikationen im Sinne der Wirtschaftlichkeit, Wartbarkeit und Markterreichbarkeit unabkömmlich.

II. EINLEITUNG

Inhaltlich betrachtet befasst sich diese Arbeit mit der Entwicklung von plattformunabhängigen, verteilten mobilen Applikationen für Smartphones. Welche Möglichkeiten stehen dazu zur Verfügung, sind diese überhaupt sinnvoll und welche Vorteile bzw. Nachteile treten auf. Dies wird anhand eines Fallbeispiels – SCADA Applikation - untersucht.

Zu Beginn werden die meist verbreiteten Betriebssysteme aufgeführt und anschließend in folgenden Bereichen verglichen: welche Programmiersprache wird verwendet, wie aufwändig ist die Erstellung bzw. Veröffentlichung eigener Apps, deren Sicherheitsaspekte werden ebenfalls betrachtet und abschließend wird ein Blick darauf geworfen welche Art von Applikationen es gibt. Native, Web oder Hybrid Applikationen werden genauer unter die Lupe genommen.

III. PLATTFORMEN

Der Handymarkt ist ein sehr großer und extrem umkämpfter Markt. Nahezu jeder Mensch hat heutzutage im westlichen Europa schon mindestens ein Handy. Viele sind aber nicht mit irgendeinem Handy zufrieden sondern stürzen sich immer wieder auf neue Produkte, welche im Monatsrhythmus erscheinen.
Zum Privathandy kommt bei vielen noch das Firmenhandy hinzu. Spitzenreiter bei Mobiltelefonen je 1000 Einwohner ist Luxemburg mit 1089,57. Knapp dahinter liegt Schweden mit 1035,53 und auf Platz 3 die Tschechische Republik mit 1010,21 Mobiltelefonen je 1000 Einwohner. Österreich liegt auf Platz 12 mit 912,99. [1]. Hierbei ist sofort zu erkennen, dass die Nachfrage an neuen Produkten immer vorhanden sein wird. Der Markt bleibt dadurch nie im Stillstand und Hersteller können mit jedem neuen Produkt die Kräfteverhältnisse zu ihren Gunsten verschieben. Viele Hersteller haben versucht in diesem Markt Fuß zu fassen und sind daran gescheitert. Vielleicht waren einerseits die Veränderungen des Marktes daran schuld, schlechtes Marketing oder die Konkurrenz mit stärkeren Produkten besser positioniert. Einer der Erfolgsgaranten bei mobilen Geräten sind die zur Verfügung stehenden Applikationen (Apps).

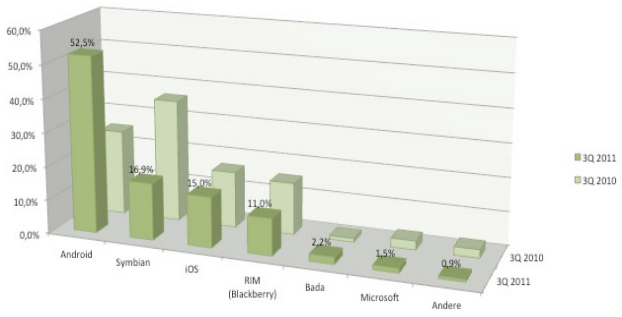

Figure1: Vergangenheit Marktverteilung[1]

Es gibt die unterschiedlichsten Betriebssysteme im Smartphone. Symbian, bada-OS, iOS, Android und Windows Phone, um die Bekanntesten zu nennen. Wir werden uns aber auf die 3 Marktführer konzentrieren. Google mit dem Android-OS, Apple mit dem iOS und Windows Phone von Microsoft. In der Grafik wird jedoch ersichtlich, dass im 3. Quartal 2011 noch ein anderes Betriebssystem unter den Top 3 zu finden war. Symbian, entwickelt von Nokia steht keine rosige Zukunft in Aussicht. Der Marktanteil ist innerhalb von 365 Tagen um mehr als die Hälfte zurückgegangen. Im Jahr 2010 lag er noch bei etwa 43% im 3. Quartal. Ein Jahr später nur mehr

bei 16,9%. Dieser Rückgang lässt sich leicht durch viel intensivere Zusammenarbeit zwischen Nokia und Microsoft erklären. Das Betriebssystem Symbian wird nicht weiter entwickelt, sondern das Unternehmen setzt auf ein neues System. Windows Phone soll in den nächsten Jahren Symbian unter den Top3 Betriebssystemen ablösen. Im direkten Vergleich liegt Microsoft mit dem Windows Phone im 3. Quartal 2011 jedoch nur mit 1,5% abgeschlagen auf Platz 6.

2012 wird das Jahr von Microsoft mit dem Windows Phone. In wenigen Jahren werden Windows-Smartphones populärer sein als Apple-Handys. Das prognostiziert die US-Marktforschungsfirma iSuppli und lobt dabei Nokias neues Handy.
Erfolg oder Niederlage vom Windows Phone hängt in den nächsten Jahren von der Zusammenarbeit zwischen Nokia und Microsoft zusammen. Alleine hatten diese 2 Firmen nur mäßigen Erfolg in den letzten Jahren. Gemeinsam wollen sie nun die Marktverteilung gehörig durcheinander wirbeln. Die Zeichen dafür stehen zurzeit sehr gut. In nur einem Jahr wurde der Marktanteil von knapp 2% auf 10% erhöht. Laut den Experten von iSuppli soll dies aber nicht das Ende des Höhenfluges sein. Bis 2015 soll ein Marktanteil von ungefähr 17% erreicht werden.[2] Die Zukunft wird zeigen, ob diese sehr ambitionierten Ziele erreicht werden können. Jedoch muss auch gesagt werden, dass Voraussagen in diesen sehr schnelllebigen Markt sehr schwierig zu treffen sind. 3 Jahre ist ein sehr langer Zeitraum in dem viel Unvorhergesehenes passieren kann. Einige Experten vergleichen, nicht unbegründet, derartige Marktprognosen mit Wettervorhersagen. 2009 wurde beispielsweise für das Windows Phone 6.5 im Jahr 2012 ein Marktanteil in der Höhe von 15% vorausgesagt. Wie wir heute wissen wurde diese Prognose bei weiten nicht erfüllt. Die nächsten Jahre werden zeigen ob die neue Kooperation zwischen Microsoft und Nokia Erfolg haben wird.

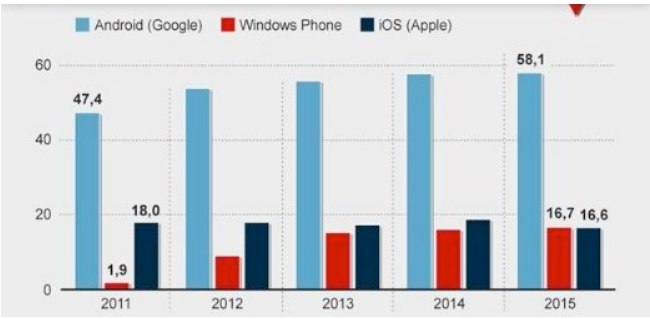

Figure2: Ausblick Marktverteilung.[2]

A. Programmiersprache[6]

Jede Applikation muss mit Hilfe einer Programmiersprache erstellt werden. Hierbei gibt es von Hersteller zu Hersteller unterschiedliche Möglichkeiten welche verwendet werden können. Bei der Programmiersprache muss man unterscheiden zwischen der Programmierung von Spielen und Applikationen. Wir werden jedoch nicht näher auf die Programmierung von Spielen eingehen.

1) Windows Phone

Beim Windows Phone gibt es mehrere Möglichkeiten um eine Applikation zu programmieren. Die Programmierung speziell mit **C#** unter dem .NET Compact Framework (.NET CF). Für die Entwicklung von Anwendungen steht eine weitere Möglichkeit in Microsoft Visual Basic .NET (VB.NET) zur Verfügung. Native Programmierung mit eMbedded Visual C++.

2) Apple iOS

Apple verwendet für seine Apps eine besondere Programmiersprache: Für die Entwicklung haben sie das Cocoa Touch Framework entwickelt. In dessen Bibliotheken, Programmierschnittstellen (APIs) und Runtimes kommt Objective-C zum Einsatz. Objective-C ist eine objektorientierte Programmiersprache.

3) Android-OS

Bei der Entwicklung von Android-Applikationen ist die üblich verwendete Programmiersprache Java. Es gibt alternativ auch die Möglichkeit mit NDK (Native Development Kit) andere Programmiersprachen verwenden zu können. Das heißt man könnte auch in C bzw. C++ für Android codieren.

B. Veröffentlichung eigener Applikationen

Grundsätzlich ist die meiste Arbeit schon vollbracht, wenn man schon Gedanken an die Veröffentlichung hat. Die fehlenden Schritte bis zur Veröffentlichung sollte kein Problem mehr darstellen wenn alles richtig und nach den Vorschriften der jeweiligen Firma programmiert wurde. Grundsätzlich muss gesagt sein, dass die notwendigen Softwareprodukte für das Programmieren gratis zu bekommen sind, jedoch wenn es um die Veröffentlichung geht Kosten für den Entwickler anfallen. Bei Microsoft kostet eine einjährige Lizenz, welche die Erlaubnis zum Veröffentlichen von Applikationen beinhaltet 99 Dollar. Für Studenten ist diese Lizenz übrigens Gratis zu bekommen. Anschließend muss die App nur verschickt werden und nach einer Prüfung und einem Validierungstest durch Microsoft, wo sichergestellt wird dass die entwickelten Apps, den Microsoft Standards der App Entwicklung, entsprechen steht dem Verkauf nichts mehr im Wege. Der Preis ist dem

Entwickler selbst überlassen, jedoch bekommt Microsoft einen Umsatzanteil von 30%.

Microsoft hat anscheinend die Eckdaten für die Kosten für die Veröffentlichung nicht zufällig gewählt. Wenn diese mit den Kosten von Apple verglichen werden, sehen wir genau dieselben Zahlen. 99 Dollar für eine einjährige Entwicklerlizenz und 30% des Umsatzes von der verkauften Applikation geht an Apple. Natürlich gibt wie bei Microsoft eine Prüfung der App, welche 7-14 Tage dauert und anschließend, bei einem bestandenen Test, kann die Applikation im App-Store runtergeladen werden.

Android-Applikationen werden über den Android-Market heruntergeladen. Mit Hilfe dieses Market können auch selbst programmierte Applikationen veröffentlicht werden. Bei der Registration fallen natürlich wieder Kosten an. Jedoch im Vergleich zur Konkurrenz sind es nur einmalig 25 Dollar. Ein zweiter, sehr großer Unterschied spiegelt sich bei der Überprüfung der Applikation auf schadhaften Inhalt wieder. Bei Microsoft und Apple wurden die Applikationen überprüft und anschließend nach erfolgreicher Prüfung zum Download freigegeben. Im Android-Market gibt es so eine Überprüfung nicht. Applikationen sind unverzüglich im Market erhältlich.

C. Sicherheitrisiko Apps?

Derjenige, der ein iPhone, Windows Phone oder Android sein eigen nennt, hat zwar ein nettes Spielzeug, welches aber ohne Applikation wenig her macht. Somit ist der Anwender gezwungen, solche kostenlosen und kostenpflichtigen Apps aus dem Store zu installieren.
Die Frage wie Applikationen für das jeweilige Betriebssystem programmiert werden können wurde im vorigen Punkt erläutert. Nun steht eine weiter Frage im Raum: Wie sicher sind Applikationen, welche aus dem jeweiligen App Store herunter geladen werden können. Apple, Microsoft bzw. Google

*1) **Apple***

Anwendungen werden einem strengen Überprüfungsprozess durch Apples "Review Team" unterzogen. Die zuständigen Mitarbeiter überprüfen dabei manuell und auch automatisiert, ob das Programm auch zur Beschreibung passt, nicht irreführend ist, Apples Richtlinien eingehalten werden und auch keine schädlichen Routinen wie beispielsweise Malware mitbringt. Die genaue Anzahl der Mitarbeiter im „Review Team“ ist leider nicht bekannt. Laut Angabe von Apple sind es wenige, aber dafür sehr gut ausgebildete Mitarbeiter. Ebenso wenig wie über die Anzahl der Mitarbeiter bekannt ist, ist der genaue Ablauf im Überprüfungsprozess eine Blackbox für Außenstehende.

*2) **Google***

Google hatte sehr lange keine wirkliche Überprüfung der Applikationen welche im Android-Market verfügbar sind.
„Bouncer“ ist der wichtigste Begriff seit der 1. Jahreshälfte 2012. Bouncer heißt im Englischen Türsteher und dies ist wohl die beste Beschreibung für Googles Sicherheitssystem. Hauptaufgabe ist das automatische Scannen des Android-Markets. Entwickler und Programmierer haben jedoch dadurch keinen zusätzlichen Zulassungsprozess, welche die Veröffentlichung der Applikation verschieben könnte. Normale Anwender, als auch Entwickler merken keinen Unterschied, da Bouncer versteckt im Hintergrund arbeitet. Die neue Sicherheitsfunktion im Android-Market scannt sowohl neue Applikationen und Anwendungen, als auch Applikationen die bereits im Android Market verfügbar sind. Des Weiteren werden auch Entwicklerkonten überprüft. Sobald eine App im Android-Market hochgeladen wurde, beginnt Bouncer mit einer Analyse auf Malware, Spyware und Trojaner. Außerdem wird das Verhalten einer jeweiligen App untersucht und mit anderen, zuvor gescannten Anwendungen verglichen, um mögliche Bedrohungen festzustellen. Die Untersuchung von Entwicklerkonten dient dazu, um "Wiederholungstäter" auszusieben und sie von einem "Comeback" über einen alternativen Account abzuhalten.

*3) **Microsoft***

Microsoft teilt die Überprüfung einer selbst programmierten Applikation in 2 Schritte auf. Als erstes wird direkt nach dem hochladen die Datei automatisch getestet. Dieser Vorgang wird XAP File Validation genannt. Anschließend wird die Applikation dem Microsoft Produktteam übergeben. Diesen 2. Schritt hat Microsoft Certification Testing genannt. Sollten diese 2 Schritte positiv abgeschlossen werden, wird die Applikation digital signiert(Signing) und veröffentlicht. Es ist schwierig über den genauen Ablauf von diesen 2 Schritten etwas heraus zu finden, da versucht wird, diese aus Sicherheitsgründen geheim zu halten.

IV. MOBILE WEB APP *VS* NATIVE APP *VS* HYBRID APP[4]

Bei den modernen Smartphones sind 2 interessante Trends zu beobachten. Erstens trifft der Slogan „Je kleiner umso besser“ nicht mehr zu. Durch die aktuellen, hochauflösenden Displays wird, z.B. die Darstellung von Videos aus dem Internet oder selbst aufgenommene immer wichtiger. Die stetig verbessernden Kameras sollten nur am Rande erwähnt werden. Der zweite Punkt ist überhaupt nicht mehr bei den heutigen Smartphones wegzudenken. Applikationen, welche aus dem Internet vom jeweiligen Anbieter herunter geladen werden können.

Welche Möglichkeiten haben modernes Smartphone heutzutage mehr als ein Handy vor 10 Jahren, abgesehen von der Weiterentwicklung der Hardware? Ohne zusätzlichen Applikationen, welche installiert werden können/müssen ist kein extremer Unterschied zu erkennen. Nun stellt sich die Frage ob diese Entwicklung in die richtige Richtung geht. Wenn man bedenkt, dass ein Smartphone viel Geld kostet und man dafür ein Gerät bekommt das nicht wirklich viel „kann“ und man für neue Feature, sprich Applikationen, zusätzlich nochmal Geld ausgeben muss fragt man sich schon ob das zu unterstützen ist.
Andererseits gibt es auch Befürworter für diese Entwicklung, denn durch den „Rohling“ welcher gekauft wird kann anschließend das Smartphone nach belieben verändert und erweitert werden. Es ist sozusagen möglich sich sein ganz eigenes, persönliches Smartphone zu kreieren.
Dadurch sollte auch ersichtlich werden, wie wichtig diese Applikationen sind und beim Kauf berücksichtigt werden sollte.
Die enorme Anzahl der downloadbaren Applikationen in den jeweiligen Stores spiegelt ebenfalls die Wichtigkeit dieser wieder.
Sowohl Softwareunternehmen als auch einzelnen Programmierern wurde durch die Applikationen ein neuer Markt eröffnet. Neue Einnahmequellen sind nicht nur durch den Verkauf von kostenpflichtigen Applikationen ermöglicht worden. Durch die Einbindung von Werbebannern, meistens in den Gratis Versionen von einer bestimmten Applikation kann ebenfalls Geld erwirtschaftet werden.

Lange Zeit war Apple mit dem App-Store an der Spitze mit den meist verfügbaren Apps. Dahinter auf Platz 2 Google und am 3. Platz Microsoft. Eine neue aktuelle Schätzung vom Jänner 2013 lässt eine Änderung an der Spitze vermuten. Google hat sich anscheinen den 1. Platz geschnappt und lässt Apple hinter sich.

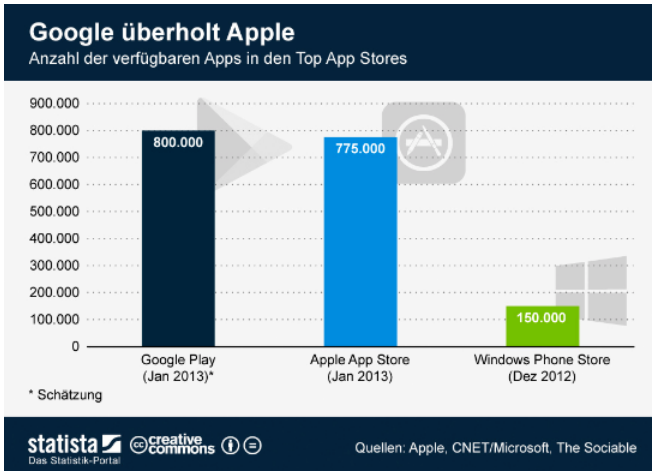

Figure3: Ausblick Marktverteilung.[12]

Native App

Sind speziell für das jeweilige Betriebssystem (Apple, Android, Windows Phone, Symbian, Blackberry, usw.) programmiert. Es hat das beste Nutzererlebnis, jedoch ist diese Applikation recht teuer in der Entwicklung. Vor allem, wenn man mehrere Betriebssysteme bedienen will, was notwendig ist, wenn man ausreichend Reichweite erzielen möchte. Native Applikationen funktionieren auch offline (außer man braucht neue Daten während der Session). Der Vertrieb erfolgt über App Store (iPhone), Android-Market (Android) usw.

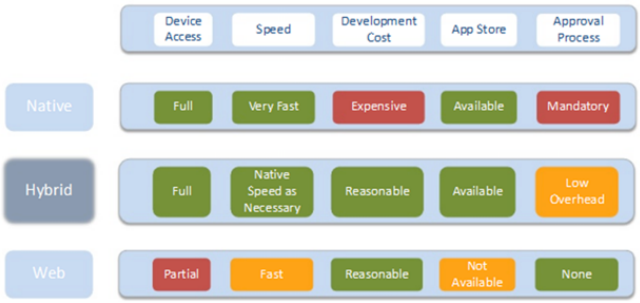

Figure4: Überblick von Applikationen[3]

Hybrid App

Diese Art der Applikationen zeichnen sich durch günstigere Entwicklungs- und Betriebskosten mit Hilfe von Frameworks aus. Das heißt, es muss nicht für jedes Betriebssystem gesondert entwickelt werden, sondern es läuft auf mehreren Betriebssystemen. “Feels Like Native” – aber dann doch nicht so ganz. Das Nutzererlebnis ist vor allem bei komplexer Interaktion nicht so gut wie bei Native Apps. Zugriff auf Hardware Funktionen, wie z.B. Kamera, Kompass etc., nur (leicht) eingeschränkt möglich. Der Vertrieb erfolgt genau wie bei den Native Applikationen durch den jeweiligen App Store.

Web App

Läuft am Browser und nutzt Webtechnologie (meist moderne Webtechnologie wie HTML5 und CSS3). Wird wie eine normale Website über den Browser angesteuert oder auch mit Lesezeichen-Icon am Springboard wie eine Native App (in iOS). Im Idealfall von einer Native App oder Hybrid App auf den ersten Blick kaum zu unterscheiden (z.B. wird das Browserfenster ausgeblendet). Geringere Entwicklungskosten und hohe Reichweite, weil Betriebssystem-übergreifend nutzbar. Jedoch schwächeres Nutzererlebnis als Native App. Der Zugriff auf Hardware Funktionen ist nur eingeschränkt möglich. Der Vertrieb über die jeweiligen App Stores ist nicht möglich.

Welche dieser 3 Möglichkeiten soll für die eigene Applikation verwendet werden? Die Antwort auf diese Frage hängt natürlich von der Art der Applikation und deren zukünftigen Verwendung ab. Daher müssen zuerst einige Gedanken an die Verwendung und Voraussetzungen getroffen werden.

- Ist der Zugriff auf Hardware Funktionen des Telefons notwendig?
- Wie häufig wird die Applikation vom User aufgerufen?
- Muss die App offline funktionieren?
- Soll mit der Applikation Geld verdient werden?
- Wie soll die Applikation vertrieben werden? App Store oder andere Plattformen?
- Zugriff auf bestimmte Gerätefunktionen notwendig?

Als letzter Punkt, ein nicht unwichtiger Punkt, sind die Entwicklungskosten. Steht mir nur ein geringes Budget zur Verfügung, muss dies natürlich auch bei der Wahl der Applikationsart berücksichtig werden.

V. SCADA SYSTEME[5]

A. Scada Software

Unter **Supervisory Control and Data Acquisition** (**SCADA**) versteht man das Überwachen und Steuern technischer Prozesse mittels eines Computer-Systems.
SCADA Systeme bieten die Möglichkeit Daten von Anlagen mit umfassenden Treibern und flexiblen Anbindungsmöglichkeiten zu anderen externen Systemen zu erfassen, visualisieren und zu steuern.

Flexibilität

Skalierbarkeit von SCADA Systemen - vom kleinen Einplatzsystem als Maschinenbedienung bis hin zum vernetzten und redundanten System.

Offenheit

Offene Konzepte erlauben die Einbindung unterschiedlichster Komponenten. Von der Automatisierungsebene bis hinauf zur Betriebsführungs- und Managementebene werden exakt zugeschnittene Lösungen ermöglicht.

Erweiterbarkeit

Moderne Anlagen sind ständigen Änderungen und neuen Anforderungen ausgesetzt. Die SCADA Software ist anpassungsfähig und wächst mit den notwendigen Erweiterungen.

B. Einzelsysteme

SCADA Systeme bieten sich für kleine lokale Systeme bis zu großen lokalen Systemen zur Überwachung und Steuerung von Anlagen an.

Diese Anlagen könnten z. B. Heizungsanlagen, Lichtersteuerung, Alarmanalgen, usw. sein.

Wird die SCADA Software autonom für Systeme eingesetzt dann besteht keine übergeordnete Sicht auf die einzelnen Systeme und damit sind auch keine (möglicherweise konsolidierten) Daten vorhanden.

VI. GLOBALE SYSTEME

A. Verteilte Systeme mit SCADA Software

Die Bildung von großen, verteilten Systems ist mit manchen SCADA Systemen gut möglich. Dabei wird ein großes System aus mehreren autonomen Systemen gebildet. Über dieses gebildete verteilte System ist es möglich alle Daten im verteilten System zu erreichen.
Ein zentral übergeordnetes System für die Überwachung kann somit implementiert werden.
Es wird dafür jedoch eine Netzwerkverbindung mit hoher Bandbreite und hoher Zuverlässigkeit benötigt.
Dies ist in geografisch verteilten Systemen nicht ohne Einsatz eines eigenen Netzwerkes nicht möglich und daher in global verteilten Systemen nicht realisierbar.

B. Wachsendes Gesamtsystem

Werden laufend neue einzelne Anlagen erstellt und zum Gesamtsystem hinzugefügt kann ein sehr großes geografisches verteiltes Netzwerk an SCADA Systemen entstehen. Als Beispiel sei hier ein Gebäudemanagement System, oder die Überwachung von Heizungsanlagen, oder Stromverbraucher in Gebäuden genannt.

C. Zentralisierung der System(daten)

Die Zusammenführung der Daten von einzelnen lokalen SCADA Systemen bietet für ein übergeordnetes Management wesentliche Vorteile:

a) Wertsteigerung durch Synergien
b) Kostenreduktion durch Gesamtoptimierungen
c) Kontrolle durch Gesamtperformanceanalysen
d) Korrelation der Daten möglich

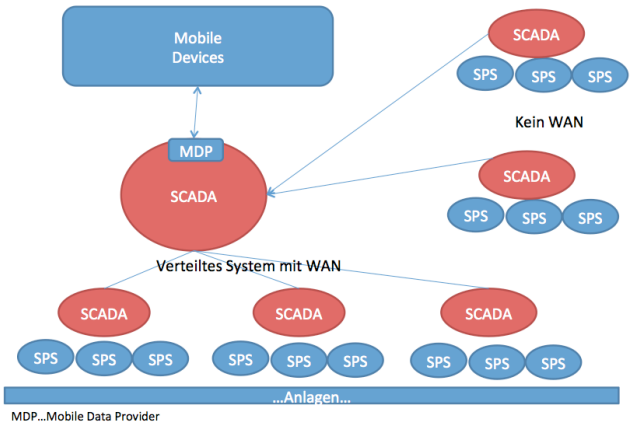

Figure5:SCADA-Zentralisierung der Systemdaten[13]

Die Zusammenführung der verteilten Daten benötigt jedoch ein weiteres System zur Datensammlung in einen zentralen Datenspeicher (Datenbank). Dadurch entsteht ein zentrales System von welchem die mobilen Clients Daten aus allen Systemen zur Verfügung gestellt bekommen können.
Bedeutet jedoch eine zentrale Abhängigkeit vom übergeordneten System. Und ein hoher Implementierungs- und Wartungsaufwand für die Datensammlung.

D. Zentralisierung in die Cloud

Für die Sammlung und Verarbeitung der Daten würde sich eine Lösung in einer Cloud anbieten.
Die Cloud kann mit der steigenden Anzahl der verteilten Systeme mitwachsen, und ist für die volatile Nutzung der mobilen Clients prädestiniert.

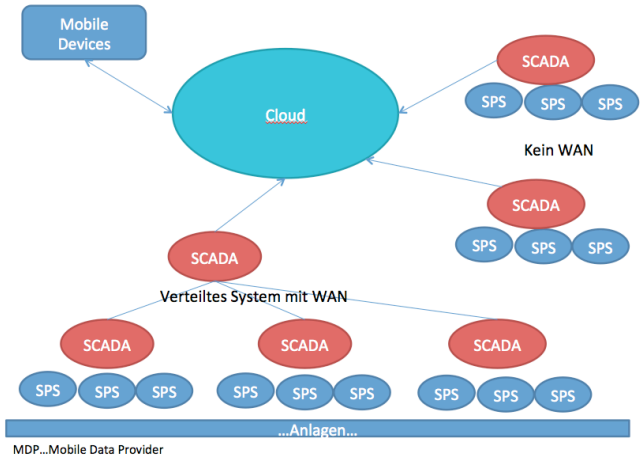

Figure6: SCADA-Zentralisierng in der Cloud[14]

E. Verteiltes System

Im Gegensatz zu einer Zusammenführung der verteilten Systeme zu einem großen globalen verteilten System ist hier der Ansatz dass sich die mobilen Applikationen autonom die Daten aus den einzelnen verteilten Systemen beschaffen.

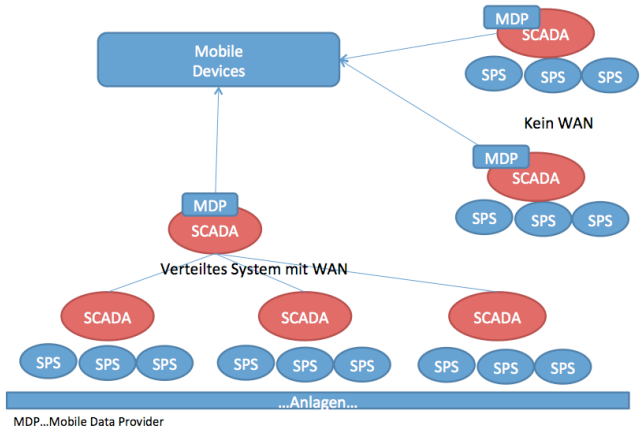

Figure7: SCADA-Verteiltes System[15]

Besteht ein System aus vielen global und geografisch verteilten Einzelsystemen bietet sich eine mobile Anwendung an um auf die Einzelsystem geografisch unabhängig zuzugreifen.

Aufgrund der möglichen weiten Distanzen zwischen den Einzelsystemen ist oft kein zentrales Datensammlungssystem vorhanden oder gar möglich, und daher eine zentrale Überwachung nicht möglich.

Es kann auch sein dass aufgrund der zentralen Abhängigkeit kein zentrales Datenmanagement System gewünscht ist.

Die zentrale Datensammlung und Datenhaltung entfällt hiermit. Ein ganzes sehr komplexes und kompliziertes System fällt damit weg, was allgemeine Vorteile sowie auch allgemeine Nachteile mit sich bringt.

Vorteile:

- Kosteneinsparung Systemerstellung
- Minimaler Wartungsaufwand / Kosten
- Hohe Verfügbarkeit, keine zentrale Abhängigkeit
- Energieeinsparung – Green IT

Nachteile:

- Verschiebung von Datenlogik in das Endgerät
- Möglicherweise beschränkter Zugriff auf gewisse Teilsysteme.
- Zentrale möglicherweise rechenintensive Datenkonsolidierung durch Einschränkungen der Endgeräte schwer möglich
- Möglicherweise redundant ausgeführte Rechenarbeiten auf mehreren Endgeräten – Energieverschwendung.

F. *Hybrides System*

In einem hybriden System wird so weit wie möglich und sinnvoll ein verteiltes System angewendet.

Für Analysen, Korrelationen, und Auswertungen welche ein mobiles Gerät aufgrund eingeschränkter Ressourcen nicht bewerkstelligen kann wird ein zentrales System geschaffen, welches die benötigten Daten von den verteilten Systemen sammelt und die rechenintensiven Algorithmen und Analysen durchführt und die (durch die Analysen komprimierten) Ergebnisdaten den Endgeräten zur Verfügung stellt.

Dieses zentrale System kann aus Sicht des Endgerätes auch als ein Teilsystem des Gesamtsystems angesehen werden. Auch bei Ausfall dieses Systems ist die Applikation auf den Endgeräten mit Einschränkungen verfügbar. Die Einschränkung beschränkt sich auf die nicht zur Verfügung stehenden Analysendaten.

Die zentralen Analysesysteme können auf mehrere Teilanalysesysteme aufgeteilt werden und ggf. auch redundant ausgelegt werden.

Weitere Überlegung wäre auch die Algorithmen verteilt zu parallelisieren. Dies könnte mit geeigneter Intelligenz auch über die mobilen Endgeräte durchgeführt werden (Grid Computing). [13]

Eine derartige Vernetzung erzeugt ein sehr komplexes Netz welches zu einer komplizierten Logik führt. Beachtet werden muss, dass Daten redundant vorhanden sind und dadurch auch von unterschiedlichen Knoten gelesen werden können oder könnten. Eine optimierte Datenbeschaffung ist ein wesentlicher Logik-Bestandteil eines solch vernetzten Systems.

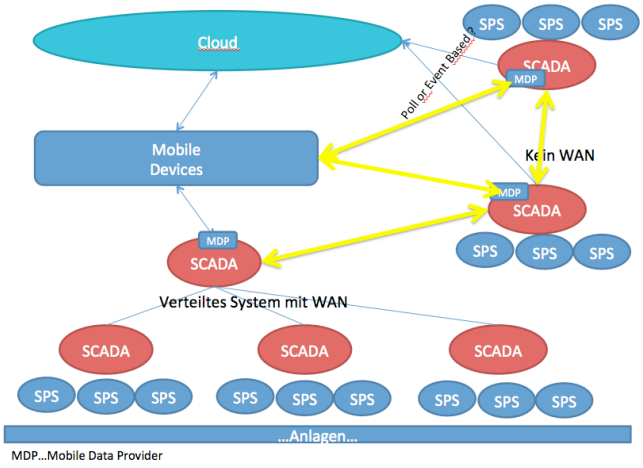

Figure8: SCADA-Hybrides System[16]

VII. MOBILE SCADA APPLIKATION

A. *Verteilte mobile Applikation*

Wie im vorherigen Abschnitt beschrieben bringt eine hybride verteilte mobile Applikation eine Reihe an Vorteilen mit sich.

Daher wird diese Architektur für die Applikation angedacht. Da diese auch die meist komplexe Architektur ist bedeutet dies eine Reihe an Anforderungen an die Gesamtapplikation und deren Einzelteile.

B. *Anforderungen an die Applikation*

Die Applikation soll auf folgenden Plattformen laufen:

- iOS – iPhone, iPad
- Android – Phone, Tablet
- Windows RT – Phone, Tablet, PC (Windows 8)

1) *Kommunikation*

Die Kommunikation zu den einzelnen Systemen soll einerseits Systemübergreifend erfolgen können, andererseits jedoch auch autonom mit nur einem System funktionieren. Die Applikation muss lauffähig sein auch wenn nicht alle Teilsystem erreichbar sind.

Die Kommunikation sollte so weit wie möglich über eine einheitliche Schnittstelle erfolgen – XMLRPC. Die Kommunikationsschicht sollte so implementiert werden dass auch die darunterliegende Kommunikationsschicht auf alternative Techniken austauschbar ist – zum Beispiel SOAP oder Protobuf von Google. Jedoch ist dies nicht ein Hauptziel.

XMLRPC[7] ist ein relativ „leichtes“ Protokoll mit relativ wenig Datenoverhead. Im Gegensatz zu SOAP ist es vor allem durch seiner einfachen und durchsichtigen Struktur hervorzuheben.

2) *Businesslogik*

Die Applikation wird aufgrund der verteilten Architektur wesentliche Logikfunktionen und Algorithmen enthalten. Systemübergreifende Analysen und Korrelationen werden am mobilen Gerät implementiert. Dabei wird es sich jedoch aufgrund der eingeschränkten Ressourcen auf mobilen Endgeräten um einfache Analysen und Korrelationen handeln.

3) *Allgemeine User Interfaces*

Die Oberfläche sollte aufgrund vieler unterschiedlicher Teilsysteme einen generischen und allgemeinen Teil enthalten welcher für jedes Teilsystem generisch verwendbar ist – bzw. dessen Verwendung ermöglicht. Dadurch bedingt werden sich die Anzeigen in diesem Bereich im tabellarischen Gestaltungsbereich bewegen.

Die generische Trenddarstellung ist auch ein generisch konfigurier- und verwendbarer Trend notwendig.

4) Spezielle User Interfaces

Für spezielle Teilsysteme (Oder Teilsystemgruppen) sollen in der Applikation auch spezielle Teilapplikation entwickelt werden können um diese Anlagen(typen) in geeigneter Weise visualisieren und bedienen zu können.

Dies umfasst Anlagenbilder und Objektbilder. Die Anlagenbilder sollten zwecks Wiederverwendbarkeit in Objektbilder gekapselt werden. Die Objektbilder sollen in anderen Teilsystemen einsetzbar sein. Voraussetzung ist dass die benötigten Daten der Objekte die gleiche Struktur haben.

Diese Oberflächen haben einen Visualisierungscharakter und können Operatoren dazu dienen um die Anlagen zu visualisieren und zu steuern. Dies soll für den Operator bestmöglich zur Verfügung gestellt werden. Es kann sich dabei um wesentliche und vor allem auch kritische Teile einer Anlage handeln. Daher ist hier auf eine bestmögliche Darstellung und auch bestmögliche Bedienbarkeit geachtet werden! Dies bringt mit sich dass auf native Funktionalitäten der jeweilig eingesetzten mobilen Plattform verwendet werden müssen!

5) Geografische Darstellungen

Eine geografische Darstellung der verteilten System soll auf einer entsprechenden Karte (Map, je nach Plattform) soll möglich sein. Hier werden die verteilten Systeme mit gewissen konfigurierbaren Anlagenbasisdaten dargestellt werden um einen bestmöglichen Überblick des Gesamtsystems zu erhalten. Ist ein System vom mobilen Endgerät nicht erreichbar muss dieser Zustand visualisiert werden.

C. Business Logik / UI Logik / UI Grafik

Die Implementierung einer Applikation der beschrieben Art verteilt sich auf:

50% Business Logik
20% UI Logik
30% UI Grafik

Diese Annahme beruht auf den Abhandlungen der vorherigen Kapitel.

VIII. FRAMEWORKS

Folgende Frameworks welche die Entwicklung von nativen Applikationen unter iOS, Android und Windows Phone unterstützen wurden in Betracht gezogen.

A. Xamarin[8]

Xamarin bietet mit MonoTouch und Mono for Android Entwicklungsplattformen mit welchen Applikationen in C# (Mono-Projekt) entwickelt werden können.

Xamarin ist eine Abspaltung von Novell welche das Mono-Projekt (http://de.wikipedia.org/wiki/Mono-Projekt) gestartet haben und hauptsächlich vorantreiben.

C# gilt als eine der innovativsten Programmiersprachen und ist in seiner Verbreitung stark steigend.

Im Jahr 2012 wurde C# die Programmiersprache des Jahres.

January headline : C# is the "language of the year"[9]

Dadurch kann einerseits die **Business Logik in C# für alle Plattformen** entwickelt, wiederverwendet und zentral gewartet werden.

Xamarin bietet in seinem Framework die nativen Funktionalitäten der jeweiligen Plattform in seinem Framework an (Wrapper Funktionalität).

Für das Userinterface bietet Xamarin alle Möglichkeiten der jeweiligen Plattform.

Der große Vorteil ist hier jedoch auch der Nachteil: **Das User Interface muss für jede Plattform separat entwickelt und gewartet werden**.

B. Qt[10]

Qt ist eine C++-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen.

Neben der Entwicklung grafischer Benutzeroberflächen bietet Qt umfangreiche Funktionen zur Internationalisierung sowie Datenbankfunktionen und XML-Unterstützung.

Qt wird insbesondere in den Bibliotheken der KDE Software Compilation 4 verwendet, welche gleichzeitig das prominenteste Vorzeigebeispiel der Klassenbibliothek darstellt.

Das User Interface wird durch die eigene Klassenbibliothek, welche auf allen Plattformen in identer Weise vorhanden und umgesetzt ist, implementiert.

Dadurch haben die Applikation auf allen verfügbaren Plattformen das gleiche Look & Feel.

Native Applikationen und Technologien der Plattform werden dabei konzeptuell bedingt nicht unterstützt.

Digia arbeitet an einer Portierung für die mobilen Platformen Android und iOS.

Die Portierungen sind noch nicht offiziell verfügbar und unterstützt.

Windows Phone spielt im Moment noch keine Rolle bei Digia. Dies könnte sich jedoch, sollte Windows Phone einen wesentlichen Marktanteil holen, ändern.

C. RhoMobile[11]

Motorola Solutions bietet mit der Software-Lösung RhoMobileSuite ein plattformübergreifendes HTML5-Entwickler-Framework.

Die Applikation läuft auf unterschiedlichen mobilen Geräten – unabhängig vom Betriebssystem und mit identischem Aussehen und gleicher Handhabung.

Das Framework basiert auf Ruby und läuft lokal auf den Endgeräten.

Die RhoMobile Suite vereint die Funktionalitäten von RhoElements, RhoConnect und RhoStudio in einer Lösung und erweitert damit den Funktionsumfang herkömmlicher Webtechnologien durch die Möglichkeit, Daten zu synchronisieren sowie on- und offline zu arbeiten.

Mit diesem Framework steht die Möglichkeit zur Verfügung Applikationen in einem einheitlichen „Look & Feel“ in nahezu jedem Betriebssystem sowie bei jeder Geräte- und Displaygröße identisch aussehen.

RhoMobile Applikationen unterstützen mobile Betriebssysteme wie Mircosoft Windows Mobile, Microsoft Windows CE und Blackberry OS sowie weit verbreitete Consumer-Betriebssysteme wie Android, Apple iOS, Symbian und Windows Phone.

Native Applikationen und Technologien der Plattform werden dabei konzeptuell bedingt nicht unterstützt.

D. Vergleich der Framworks

RhoMobile ist die allgemeinste Möglichkeit für die Entwicklung einer Applikation unter unterschiedlichen Plattformen. Die verwendete Programmiersprache ist zurzeit ein Exote. Durch die Verwendung auf HTML5 werden die Applikationen naturgemäß nicht nativ wie es sich Benutzer wünschen.

QT bietet ein Framework mit welchem Applikationen gänzlich plattformunabhängig und erstellt werden können. Dadurch auch ein neutralisiertes Layout erhalten – auch wenn dies am Desktop Windows/Linux nicht leicht zu unterscheiden ist.

Wie dies auf den mobilen Plattformen aussehen wird ist noch abzuwarten (da die Portierung erst in der Umsetzung ist und man daher noch keine Referenzwerte hat). Grundsätzlich unterscheiden sich die Bedienphilosophien von z.B. iPhone und Android mehr als die von Desktop Betriebssystemen wie z.B. Windows und Linux.

Die Verwendung von nativen Möglichkeiten der Plattform ist möglich, jedoch schwierig in der Einbindung da keine vorgegebenen Schnittstellen vorhanden sind, und diese selbst erstellt werden müssen.

C++ ist eine solide und weit verbreitete Programmiersprache, jedoch für eine schnelle und stabile Applikationsentwicklung nicht mehr „State oft the Art“ ist.

Xamarin bietet die Möglichkeit native Applikation für die gewählten Plattformen zu erstellen. Die Applikationen können bzw. müssen im User Interface Bereich für die jeweilige Plattform individuell erstellt werden.

Dies kann grundsätzlich als Nachteil für die Wartbarkeit, jedoch auch als Vorteil für die Unterstützung der nativen und plattformspezifischen Möglichkeiten und deren Look&Feel angesehen werden.

C# als Programmiersprache bietet eine Reihe an Vorteilen, und wird aus heutiger Sicht seine Marktpositionierung weiter ausbauen.

- Managed Code
- .Net Framework
- Distribution Support
- Support for parallel programming
- Moderne Programmiersprache
- Leichter „Aufstieg“ von C++ auf C#
- Plattformunabhängkeit durch Mono
- Sprachneutral durch die CLR (entstehende Libraries könnten auch von C++ oder anderen .Net Programmiersprachen genutzt werden)
- Sicherheit in der CLR

Tabellarischer Vergleich:

	Xamarin	QT	RhoMobile
License	Commercial	Commercial	Open Source
Type	Framework	Framework	Platform
Language	C#	C++	Ruby
Common/Server	C#	C++	No
Platform independed	Yes (Mono)	Yes	No
Supported Frameworks	.Net	No	No
Distribution Support	Yes (.Net Remoting, ...)	No	No
Safe Programming	Yes	No	No
Security	High	No	No
Application			
User Interface	Nativ	Generic	HTML5
iOS	Yes	under Construction	Yes
Android	Yes	Yes	Yes
Windows	Yes	-	Yes
Offline Available	Yes	Yes	Partly
Nativ Look&Feel	Yes	No	No
Nativ Features	Yes	Partly	Partly
New Features Support	Fast	Questionable	No

IX. ZUSAMMENFASSUNG

Der allgemeine Drang von HTML zu nativen Applikationen wird durch Smartphones gefördert und gebildet. Die Bedienung ist einfacher und auch Offline möglich ist.

Microsoft geht hier auch den Weg dass Apps bis zum Desktop durchgereicht werden – mit Windows 8 werden Apps sowohl am Smartphone als auch am Desktop verwendet. Der Grundgedanke hierbei ist durchaus nachvollziehbar wenn man Apps mit einem Web-Auftritt vergleicht, oft ist die Bedienung von Apps am Handy einfacher und besser als die Webversion am Desktop.

Daher ist eine native Applikation für das Smartphone ein wesentlicher Erfolgsgarant für eine Applikation. Werden auch die nativen Möglichkeiten des jeweiligen Smartphones unterstützt bringt das einen wesentlich Vorteil in den Möglichkeiten zur Bildung einer Applikation.

Natürlich muss beachtet werden dass auf Grund dessen kein einheitlicher Code für alle Plattformen entwickeln lässt. Hat man jedoch eine einheitliche Programmiersprache dann lässt sich auch dieses Manko auf eine mögliches Minimum reduzieren und kapseln.

Eine einheitliche Programmiersprache von der Clientapplikation bis zur Serverapplikation hat weitere wesentlich Synergieeffekte für die Gesamtapplikation / das Gesamtsystem – Wiederverwendung von
Code, Klassen, Datenstrukturen.

Know How der schaffenden Menschen (Programmierer, Konzeptionisten, ...) kann durch eine einheitliche Programmiersprache übergreifend und allgemein eingesetzt werden. Gleiche Sprache ist ein wesentlicher Erfolg, auch in der Kommunikation!

Bietet die eingesetzte Programmiersprache solide, weitreichende, und unterstützte Bibliotheken (Frameworks) als Basis, so kann auf eine solide Basis zurückgegriffen werden welche stabil und weiter entwickelt wird.

Zusammengefasst bietet das Xamarin Framework all diese beschriebenen Vorteile und kann somit als zukunftsreiche Technologie bezeichnet werden.

QT ist ebenfalls ein potentieller Kandidat und geht in eine ähnliche Richtung mit dem Vorteil echte native (nicht managed Code) zu bieten und hat vor allem auch im Desktop Bereich eine Grundsolide und weit verbreitete Marktpräsenz. Und bietet wirkliche Portierbarkeit plattformunabhängig.

Die eingesetzte Programmiersprache ist jedoch für die aktuelle Zeit der IT eine „Veraltete“, grundsätzlicher Vorteil von C# gegenüber C++ ist ein managed Code und ein umfangreiches .Net Framework. Beides führt zu einer schnelleren Erstellung von stabilen Applikation.

[1]www.welt-in Zahlen.de/laendervergleich.phtml?indicator=111 , last update: 01.04.2007, 02.12.2012
[2]www.isuppli.com/Mobile-and-Wireless-Communications/News/Pages/Lumia-900-Introduction-to-Trigger-Smartphone-Renaissance-for-Nokia-and-Microsoft.aspx, 03.12.2012
[3]http://econsultancy.com/at/blog/7832-the-fight-gets-technical-mobile-apps-vs-mobile-sites, 05.12.2012
[4]Wie viel kostet die Entwicklung von Apps?
Andreas Blüml (Autor), Andreas Frank (Autor); Seite 12
[5]Techno Security's Guide to Securing SCADA
Jack Wiles (Autor); Seite 62
[6]Der App-Entwickler-Crashkurs für Android, iOS und Windows Phone. Christian Immler (Autor); Seite 9
[7]XML-RPC Agents for Distributed Scientific Computing
Robert van Engelen (Autor), Kyle Gallivan (Autor), Gunjan Gupta (Autor), George Cybenko(Autor);
[8] Professional Mobile Application Development
Jeff McWherter (Autor), Scott Gowell (Autor); Seite 348
[9]https://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language; 02.01.2013
[10]qt.digia.com; 30.12.2012
[11] http://rhomobile.com/products/rhodes/; 02.01.2013
[12] http://de.statista.com/themen/882/apps-app-stores/infografik/810/anzahl-der-verfuegbaren-apps-in-den-top-app-stores/; 08.01.2013
[13]A survey of mobile cloud computing: architecture, applications, and approaches
Hoang T. Dinh, Chonho Lee, Dusit Niyato and Ping Wang
School of Computer Engineering, Nanyang Technological University (NTU), Singapore; 20.12.2012