Projektdokumentation

**Projekt42**

C++/Qt Implementierung eines Prototypen

für Verkaufsoberflächen

ETB Electronic Team

Beratungs- und Vertriebs GmbH

Grabauerstrasse 53

21493 Schwarzenbek

Durchführung

Jan Schneidereit

Fachinformatiker / Anwendungsentwicklung

letzte Änderung 10.05.2011

1. 1 Projektdefinition
   1. 1.1 Projektauftrag

Die ETB Electronic Team Beratungs- und Vertriebs GmbH biete eine Vielzahl von Lösungen im Bereich der Telemetrie, sowie Mensch-Maschine-Systeme für Verkaufsautomaten an. Die Aufgabe dieses Projektes beinhaltet die Proof of Concept Implementierung einer neuen POS/Kiosk Oberfläche für Vending Automaten.

Mittels dieser Studie soll nachgewiesen werden, in welchem Umfang sich eine alternative Lösung für die bereits bestehende Verkaufsoberfläche auf kleinere Hard- und Software Systeme portieren lässt.

Das Projekt selbst besteht aus der Kernsoftware, die auf einem Embedded Linux System aufgesetzt wird. Die Kernsoftware selbst wird in ein Backend für Programmlogiken und Kommunikationsschnittstellen, sowie ein Frontend für die Grafische Präsentation von Funktionalität an den Benutzer unterteilt.

Das Backend stellt die Grundlegende Schnittstelle zwischen Grafischer Oberfläche und der Hardware dar. Die Kommunikation mit der Maschine ist nicht Bestandteil dieses Projektes. Über die dafür nötige Implementierung wird erst nach der Beurteilung des Prototypen durch den Auftraggeber entschieden.

Das Frontend soll mindestens 3 verschiedene Oberflächen beinhalten, die die verschiedene Zustände einer Verkaufsoberfläche repräsentieren. Unter anderem eine Produktauswahl, eine Bedienoberfläche zum variieren eines Produktes, sowie ein Mediabildschirm. Weitere Oberflächen sind optional und dienen ausschließlich der Vervollständigung des Prototypen.

Diese Oberflächen sollen auf zwei verschiedene Arten implementiert werden können.

Zum einen speziell für diese Software entwickelte Oberflächen, die direkt in einer dafür vorgesehen Sprache geschrieben werden. Zum anderen Dynamisch generierte Oberflächen, die aus dem bestehenden Backend für solche Oberflächen generiert werden. Dies erfordert eine Anpassung des Backends dieser Software an das Format der Oberflächen aus dem bestehenden System.

Die Bedienung des Systems soll wie im Äquivalent auf Touchscreen Bedienung optimiert sein.

* 1. 1.2 Projektumfeld

Das Projekt gehört thematisch zur Abteilung für die Entwicklung von Lösungen rund um Verkaufsautomaten. In diesem Umfeld werden Prototypen und Serienreife Produkte für Großkunden entwickelt. Produkte, die in diesem Umfeld vertrieben werden, sind für den industriellen Einsatz konzipiert, daher werden hier Systeme für den Dauerbetrieb ausgelegt. Da es sich um ein Proof of Concept handelt wird erst nach Projektabschluss über eine Integration des Projektes in die bestehende Entwicklung von POS (Point of Sale) Systemen entschieden.

* 1. 1.3 Projektabgrenzung

Die Software, die in diesem Projekt umgesetzt wird, ist zum größten Teil unabhängig von bereits bestehenden Systemen. Die zugrunde liegende Technologie wurde bisher nicht eingesetzt und erfordert eine gänzliche Neukonzipierung. Bereits zur Verfügung gestellte Schnittstellen, sollen im späteren Verlauf ohne große Anpassungen weiter genutzt werden können.

**2 Projektplanung**

**2.1 Analyse**

**2.1.1 Ist-Analyse**

Die bisherige Implementierung der Verkaufsoberflächen erfolge mit Webtechnologien, dies erfordert das zusammenspiel mehrerer verschiedener Softwarepakete. Die Visualisierung der Benutzerschnittstellen so wie der großteil der Programmlogiken erfolgt interpretativ.

* + 1. 2.1.2 Ist-Kritik

Eine Problematik des Webbasierten Ansatzen zur realisierung der Verkaufsoberflächen ist der hohe Bedarf an Systemressourcen. Das resultiert darin, das für hohe Anforderungen an das User-Interface immer stärkere Rechensysteme benötigt werden, die teilweise sehr groß dimensioniert werden müssen um den Anforderungen der Software gerecht zu werden.

Zudem mangelt es den teilwese zweckentfremdeten Komponenten an flexibilität, was bereits in einer frühen phase der entwicklung von neuen Systemen zu Problemen oder Verzögerungen fehlt.

**2.1.3 Soll-Konzept**

Ziel der Entwicklung dieses Prototypen ist es herauszufinden, ob die Bereitstellung der Verkaufsoberflächen mit einem technisch geringeren Aufwand zu realisieren ist. Es soll eine spezialisierte Software entwickelt werden, die für genau diese eine Aufgabe konzipiert wurde, um somit die Anforderungen an das Hostsystem zu verringern und sowohl die Skalierbarkeit als auch die Erweiterbarkeit gegenüber des bisherigen Ansatzen zu verbessern.

Aufgrund der Unterschiedlichen Einsatzgebiete soll es möglich sein, die Software für Bereiche mit geringeren oder höheren Anforderungen anzupassen.

* + 1. 2.1.4 Anforderungen

**Der Prototyp soll später auf Embeddedsystemen zum Einsatz kommen, muss deshalb mit möglichst wenig Speicher und Rechenleistung auskommen. Da Verkaufsoberfläche in der Regel für eine breite Masse an Benutzern vorgesehen ist sollte gewährleistet sein, dass dem Benutzer gegenüber keine Einschränkungen im Bezug auf** ***Software-Ergonomie* auftreten. Das beinhaltet mehrere Punkte wie ständige Verfügbarkeit, unmittelbare Verständlichkeit der Benutzerführung oder der Aufgabenangemessenheit.**

**Zu Testzwecken wird ein Beispiel für die Menüführung eines Kaffeeautomaten entwickelt, die es einem Benutzer ermöglichen soll möglichst intuitiv und schnell an das gewünschte Produkt zu gelangen.**

**Eine häufige Anforderung an moderne Benutzerschnittstellen ist die Multimediafähigkeit. Als Erweiterung wird deshalb zusätzlich ein Mediabildschirm integriert, um herauszufinden wie geeignet die verwendete Technologie im Bezug auf Mediale Inhalte ist.**

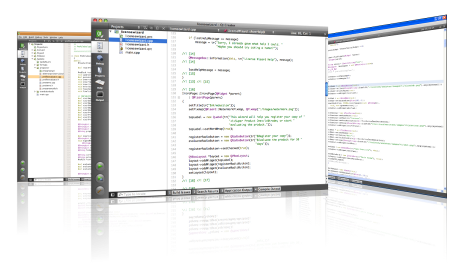
* 1. 2.2 Ressourcenplanung
     1. 2.2.1 Zeitplanung

|  |  |  |
| --- | --- | --- |
| Zeitplanung | | |
| Projektphase | Beschreibung | Zeitaufwand in Std. |
| Analyse | Herausarbeiten der Ist-Analyse und des Soll-konzeptes  Ausarbeitung Anfoderungsprofil | 4 |
| Entwurf | Entwurf der Kernsoftware des Prototypen in UML | 5 |
| Implementierung | Programmieren der Kernsoftware  und der Benutzerschnittstellen | 43 |
| Test | Test des Prototypen und test der Implementierung schnittstelle Backoffice | 2 |
| Dokumentation | Schreiben der Dokumentation und  aller dazugehörigen Dokumente | 15 |

* + 1. 2.2.2 Kostenplanung

1. 3 Realisierung
   1. 3.1 Die Entwicklungsumgebung

Die Kernsoftware wird in einer C++/Qt basierten Umgebung entwickelt. Als IDE wird der QtCreator v. 2.2 verwendet, einem C++ orientierten Editor, der speziell für die Verwendung des Qt Frameworks entwickelt wird. Diese IDE ermöglicht ein sehr effizientes Arbeiten durch zahlreiche Funktionen wie Syntax Highlighting, Code Completion, Semantic Navigation und GUI Designer für sowohl Widget basierte Oberfläche als auch QML basierte Oberfläche.



* + 1. 3.1.1 Programmiersprache

Die benutzte Programmiersprache für dieses Projekt ist C++.

Der verwendete Compiler unter Windows ist MinGW. Eine Portierung der GCC (GNU Compiler Collection) für Windows Systeme. Unter dem Linux Entwicklungssystem wird direkt die GCC in der Version 4.5.0 verwendet.

Zu Teilen wird auch eine Javascript Anbindung genutzt. Diese wird durch die QML Umbegung bereitgestellt. So kann man Layoutspezifische logiken auf einer für Designer leichter Verständlichen Ebene umsetzen.

* + 1. 3.1.2 Framework

Als Framework wird das C++ basierte Qt Framework genutzt. Dieses Framework biete umfangreiche Bibliotheken für das Erstellen grafischer Oberflächen, ein intuitives File-Handling, die Implementierung OpenGL basierter Software, sowie die Anbindung an Netzwerke, Datenbanken und vieles mehr. Die grafische Schnittstelle dieses Projektes wird zum größten Teil mit einer neuen Technik namens "Qt Quick" umgesetzt. Diese Technologie beinhaltet eine deklarative Auszeichnungssprache "QML", die es ermöglicht auf sehr einfache Weise grafische Oberflächen zu beschreiben, dadurch sind auch Designer ohne C++ Kenntnisse in der Lage Oberflächen zu gestallten. Außerdem beinhaltet "Qt Quick" die "QtDeclarative" Module. Diese Bibliotheken dienen dazu, die mit QML beschriebenen Oberflächen auf C++ Ebene umzusetzen und zu rendern. Dieses Verfahren führt zu verbesserter Performance was sich gerade auf Target Systemen mit wenigen Ressourcen bezahlt macht. Qt Quick wurde aus diesem Grund von Nokia initialisiert und stark unterstützt, um Mobile Systeme wie Smartphones mit Qt Quick auszustatten.

Das Qt Framework wurde Ursprünglich von Trolltech entwickelt und wurde später von Nokia übernommen. Es steht in verschiedenen Lizenzmodellen zur Verfügung und wird im Rahmen dieses Projektes unter der LGPL Lizenz verwendet.Das ermöglicht den Kostenlosen Einsatz des Frameworks.Gegenüber der GPL ist es außerdem möglich die entwickelte Software als Closed-Source Software zu vertreiben.

Das Projekt wurde mit der Qt Version 4.7.2 initialisiert und während der Entwicklungsphase auf die Version 4.7.3 Portiert.

* + 1. 3.1.3 Versionsverwaltung

A description...

Für die Versionsverwaltung kommt Git zum einsatz. Diese relativ neue und moderne Versionsverwaltungsystem bietet umfangreiche Möglichkeiten und ist frei verfügbar. Die Verwaltung der Repositorien geschieht dezentral, das heißt es gibt kein zentrales Repository auf das die Entwickler zugreifen müssen, sondern jeder Entwickler hat eine Kopie des kompletten Projektrepositories auf seinem Arbeitsrechner. So kann jeder Entwickler auch ohne Netzzugang über die volle Funktionalität verfügen. Es gibt aber meist ein „offizielles Repository“, das den aktuellsten Stand eines Projektes representiert.

Die Vorteile der Versionsverwaltung sind vielschichtig, zum einem wird der komplette Code auf konsistenz überwacht, so dass bei gleichzeitigen Änderungen der Entwickler dafür verantwortlich ist evtl. Konflikte gleich zu lösen. Zum anderen werden sämtliche Änderungen Protokolliert, dadurch kann jeder zeit eingesehen werden, wann und von wem eine Änderung vorgenommen wurde. Zudem kann für den Fall das dies nötig wird auf einen älteren Projektstand zurück gekehrt werden.

Das Projekt wurde auf GitHub gesichert, einem Server für sogenannte „offizielle Repositorien“. Hier liegt also der aktuellste Stand des Projektes.

Git kam in der Version 1.7.1 zum Einsatz.

* 1. 3.2 Schnittstellendefinition

Aufgrund des Prototypenstatus dieses Projektes wurden weitestgehend auf Schnittstellen verzichtet, da diese erst dann umgesetzt würden, wenn eine Entscheidung über die Tauglichkeit des technologischen Ansatzes dieser Software gefällt wurde.

Als einzige Schnittstelle ist hier die Anbindung des XML Layout Interpreters für das Backoffice zu erwähnen. Dieser wurde gefordert um herauszufinden, ob sich die Software dafür empfehlen kann, eine bereits existierende Schnittstelle für das dynamische generieren von Layout Templates zu nutzen.

* 1. **3.3 Entwicklungsphasen**
  2. 3.3.1 Softwaredesign

**Zeitraum: 07.04.11**

**Zeitaufwand: 6 Stunden**

**In der Entwurfsphase wurde das Konzept der Software einmal als Klassendiagramm Definiert. Dieses UML Diagramm dient später als Leitfaden für die Implementierung und wird anschließend an mögliche Änderungen bei der Programmierung angepasst.**

**Zur Designphase gehörte auch die Absprache eines *Mock-up* für ein QML basiertes Layout. Dieses Mock-up wurde in dieser Phase nach den Gesichtpunkten bezüglich der Anforderung an die Benutzeroberfläche gestalltet und später in der Implementierungsphase in vollem umfang Programmiert.**

**3.3.2 Implementierung**

**Die Implementierung erfolgte aufgrund von Zeitlichen Konflikten mit anderen Projekten in nicht zusammenhängenden Zeiträumen. Zudem wurden die unterschiedlichen Teile des Prototypen nicht aufeinander folgend Entwickelt sondern parallel. So das zeitweise sowohl am Backend als auch am Frontend gleichzeitig gearbeitet wurde.**

* 1. 3.3.3 Backend

Zeitraum: vom 18.04.11 – 20.04.11

Zeitaufwand: ca. 10 Stunden

Entwicklung eines eigenständigen Kernprozesses, der sämtliche Aufgaben vereint. Dazu gehörte die Integration eines Viewports, der für die grafische Darstellung des Interfaces zuständig ist.

Zeitraum: vom 09.05.11 – 13.05.11

Zeitaufwand: 14 Stunden

Implementierung aller nötigen Klassen und Typen für die Kommunikation zwischen dem Frontend und dem Backend. Sowie ein Logger, der sämtliche wichtigen Ereignisse in entsprechenden Log dateien festhält. Zusätzlich wurde hier auch der Parser für die XML basierten Konfigurationsdateien geschrieben, die für alternative visuelle Layouts genutzt werden könnten.

Die Implementierung dieses XML Parsers wurde nur in begrenztem Umfang übernommen, da es hierbei nur um den Nachweis der Machbarkeit ging.

* 1. 3.3.3 Frontend

Zeitraum: vom 26.04.11 – 28.04.11

Zeitaufwand: ca. 6 Stunden

Zum einen soll das Layout per XML definiert werden können. Der hierfür benötigte Parser wurde allerdings erst zu einem späteren Zeitpunkt geschrieben. Hier wurden zu testzwecken vorerst die Datentypen definiert.

Zum anderen wurde eine Grafische Benutzerschnittstelle in QML geschrieben, die demonstrieren soll, was technisch möglich wäre. Sie soll aufzeigen, was zum momentanen Zeitpunk mit dieser Software darstellbar ist, um später bei einer Entscheidung zu zeigen, wo stärken und schwächen liegen.

Zeitraum: vom 16.05.11 – 19.05.11

Zeitaufwand: 11 Stunden

Vervollständigung des QML und XML Interfaces. Beide Ansätze waren zu diesem Zeitpunkt nur in einem Alpha Stadium vorhanden und wurden hier nochmal angepasst und auf ihren entgültigen Stand gebracht.

Dies geschah erst jetzt, das hierzu noch bestimmte Funktionalitäten benötigt wurden, die erst jetzt zur verfügung standen.

* + 1. 3.3.5 Schnittstellen

Die Entwicklung der unidirektionalen XML Schnitstelle vom Backoffice zur Bereitstellung von XML basierten Layouts wurde in der zweiten Frontend Entwicklungsphase durchgeführt.

* 1. 3.4 Konfiguration des Zielsystems
     1. 3.4.1 Deployment
     2. 3.4.2 Testphase

1. 4 Projektbewertung
   1. 4.1 Funktionalität der Implementierung
   2. 4.2 Vergleich mit Alternativsystem

Für die Entwicklung solch einer Anwendung ist nicht direkt an eine Entwicklungsumgebung gebunden. Es ist also nicht zwingend Notwendig Qt dafür zu nutzen. Es erschien mir aber aus verschiedenen Gesichtspunkten die beste Wahl zu sein.

Es gibt auch andere Frameworks für die Entwicklung von Anwendungen mit hochwertigeren Benutzerschnittstellen.

Wie JavaFX, Flash oder Silverlight, die zum teil alle eine Problem mit sich bringen. Alle diese Techniken wurde in erster Linie für die programmierung von Rich Internet Applications entwickelt.

Flash ließe sich zwar über die Adobe Air Laufzeitumgebung als Stand alone Anwendung auslegen. Ein Problem an Flash ist aber die Performance. So wurde Flash bereits genutzt, um als Browserplugin, die Benutzerschnittstelle zu representieren. Was zu teilweise massiven Ansprüchen an die Rechensysteme führte.

Silverlight bringt zusätzlich das Problem mit, dass es nicht nativ auf Linux basierten Systemen läuft, sondern nur über eine Portierung namens „Moonlight“ auf solchen Systemen läuft. Da es bei solchen Ansätze und zu Problemen kommt erschien mir das ebenfalls als wenig empfehlenswert.

Was in diesem Fall für Qt/QML als Basis der Entwicklung spricht ist die Tatsache, dass Qt auf C++ basiert und die Programme in nativen Maschinencode übersetzt werden. Das bringt erhebliche Vorteile für die Performance der Software, da große Bestandteile direkt ausgeführt werden, anstatt dass diese erst interpretiert werden müssen. QML selbst ist da differenziert zu betrachten.

QML selbst ist eine Deklarative Sprache, die nicht in Maschinencode übersetzt wird. Das .qml Format wird von der Qdeclarative Laufzeitumgebung interpretiert, ist deshalb nicht so performant die der C++ kern, bietet dafür aber die beste schnittstelle zwischen einer schnellen Kernsoftware und dem Benutzer. QML ist leicht zu erlernen und kann auch von Webdesignern bedient werden, so wird es ermöglicht, die Benutzerschnittstelle, ohne Notwendige Änderungen am Kern ändern zu können.

Da sich „Qt Quick“ also die QML Laufzeitumgebung noch in den Kinderschuhen befindet, gibt es natürlich noch mängel, die es zu beheben gilt. Zum einen wurde die Technik eher für mobile Plattformen entwickelt, um dort einfachere Menüstrukturen abbilden zu können. In einem Anwendungsfall wie diesem, in dem das Hauptaugenmerk auf der Bereistellung möglichst flüssiger Oberflächen liegt, sind die Anforderungen wesentlich höher. Aus diesem Grund kann es vorkommen, dass auf schwächeren Rechensystemen zu spürbaren Verzögerungen beim Rendern kommt. Das Problem ist aber seitens der Entwickler bei Nokia schon bekannt und wird bereits mit entsprechenden Weiterentwicklungen in naher Zukunft behoben werden. Unter anderen soll der QML engine ein „***Scene graph“ zugrunde gelegt werden. Was die Performance erheblich steigern soll.***

* 1. 4.3 Einhaltung der Planung

Die eingeplante Entwicklungszeit von 43 Stunden konnte nicht eingehalten werden. Aufgrund der vorher nicht abschätzbaren Komplexität der integration der XML Modul typen wurde hier etwas mehr zeit benötigt. Um sowohl das parsing der Module deren Speicherung und deren Eventhandling zu realisieren.

Zudem gab es einige Schwierigkeiten bezüglich der Zugriffe auf Globale Javascript Variablen in der QML Umgebung, aufgrund von differenzierten Kontexten, die vorher nicht abzusehen waren, die zusätzlich zu leichten Verzögerungen führten.

Die Zeit für die Entwurfsphase wurde leicht überschritten, um ein möglichst Vollständiges Klassendiagramm entwickeln zu können, was dann die Implementierung der Klassen etwas vereinfachte.

Die Testphase wurde aufgrund der im Prototyp sehr Begrentzen Funktionalität allg. kurz gehalten. Aufgrund von intensiveren Tests bereits während der Entwicklungsphase, konnte hier zusätzlich noch Zeit eingespart werden.

|  |  |  |
| --- | --- | --- |
| **Zeitplanung** | | |
| **Projektphase** | **Beschreibung** | **Zeitaufwand Soll / Ist** |
| **Analyse** | **Herausarbeiten der Ist-Analyse und des Soll-konzeptes**  **Ausarbeitung Anfoderungsprofil** | **4 / 3** |
| **Entwurf** | **Entwurf der Kernsoftware des Prototypen in UML** | **5 / 6** |
| **Implementierung** | **Programmieren der Kernsoftware**  **und der Benutzerschnittstellen** | **43 / 47** |
| **Test** | **Test des Prototypen und test der Implementierung schnittstelle Backoffice** | **2 / 1** |
| **Dokumentation** | **Schreiben der Dokumentation und**  **aller dazugehörigen Dokumente** | **15 / 15** |
| **Gesamt** |  | **69 / 72** |

2. 5 Glossar

**Backend – zu Deutsch „hinteres“ oder „unteres“ Ende. Das Backend beschreibt in diesem Fall die Schnittstelle des System zum darunter liegenden Hardwaresystem.**

**Während das Frontend für die Kommunikation mit dem Benutzer entwickelt wird ist das Backend ausschließlich für die Kommunikation mit dem Automaten bzw. dem Hostsystem zuständig.**

**Backoffice – als Backoffice wird in diesem Kontext die Serverseitige Schnittstelle des Unternehmens nach Außen bezeichnet. Hier werden verschiedene Daten bereit gestellt, die von unterschiedlichen Klienten genutzt werden können.**

**Closed-Source – Im gegensatz zu OpenSource Software, also Software, zu der der Quelltext offen gelegt werden muss, beinhaltet der Begriff Closed-Source ausschließlich das Ausliefern der Software im Binärformat. Also eine bereits für ein bestimmtes System in Maschinencode übersetzes Format. Das sich weder gut lesen noch ändern lässt. Dies sollte eine Anforderung für *Proprietäre Software sein.***

**Code Completion -** Ist eine Technik, die dazu dient dem Programmierer Vorschläge zu machen, welche Funktionen oder Variablen er verwenden kann. Die IDE oder der Editor, der diese Funktion anbietet, analysiert was der Programmierer grade schreibt und macht seinerseits Vorschläge, zu dem was der Programmierer meinen Könnte, das spart Zeit durch das Vermeiden von Fehlern, die auf möglichen Schreibfehlern basieren.

**Compiler** - Ein Compiler ist ein Programm, das den vom Programmierer in einer bestimmten Sprache geschriebenen Quellcode in z.B. Maschinen Code umwandelt.

Das ist dazu nötig, damit die Zielmaschine das Programm versteht und ausführen kann.

Das Compiling erfolgt Zielsystem abhängig. Der Compiler muss also wissen, als welcher Art von Hardwarearchitektur das Programm später laufen soll, um so in einen passenden Maschinencode zu übersetzen.

**Debugger** - sind Werkzeuge für die Softwareentwicklung, die es ermöglichen Fehler in Programmen zur Laufzeit zu finden oder das Programm zur Laufzeit zu diagnostizieren.

**Embedded – zu Deutsch „eingebettet“ bezieht sich auf Verschiedene Bereiche.**

**In diesem Fall bezieht sich der Begriff auf das portieren eines Softwaresystems auf Embedded Computer, also möglichst schmale Rechensysteme, die gerade so viel leistung aufbringen um eine definierte Aufgabe abwickeln zu können.**

**Framework** - Als Frameworks sind Sammlungen von Funktionen also Bibliotheken zu verstehen, die dem Programmierer bereits festgelegte Funktionen zur Verfügung stellen.

Durch die Verwendung von Frameworks spart der Entwickler sehr viel Zeit, die er sonst damit verbringen müsste, Funktionen selbst zu schreiben, die häufig benötigt werden.

**Frontend – bezeichnet die „vordere“ Schicht eines Softwaresystems.**

**Das Frontend kann als Schnittstelle zwischen dem Benutzer und dem System angesehen werden. Das Frontend hat die Aufgabe dem Benutzer eine möglichst intuitive Oberfläche bereit zu stellen, die das bedienen eines Sysems vereinfacht und somit eine möglichst hohe akzeptanz erreicht.**

**GUI** - "Graphical User Interface"

Der Englische Begriff für grafische Oberflächen. Die dem Benutzer einer Software, einen möglichst effektiven Einsatz von Software ermöglichen soll.

**Hostsystem** – Das Hostsystem bezeichnet die Hardwareplattform also das Rechensystem auf dem die Software laufen soll.

**IDE** - Integrated Development Environment

Ist eine Sammlung von Programmen in einem Programmpaket oder Editor, der es dem Entwickler ermöglicht alle zum Entwicklungsprozess benötigten Werkzeuge in einer Oberfläche zu vereinen. Dies beinhaltet i.d.R einen Editor, Compiler, Debugger, Syntax Highlighting etc.

**Kiosk – bezeichnet die Verkaufsstelle bzw. im Kontext machner Software einen Modus der darauf ausgelegt ist auf speziellen Maschinen zu laufen. Im „Kioskmodus“ hat der Anwender nur noch eingeschränkte Möglichkeiten, kann z.B. das Programm nicht schließen oder andere Anweisungen ausführen. Das dient meist dazu eine nahtlose grafische oberfläche anzubieten.**

**MinGW** - Minimalist GNU for Windows ist die Portierung der GNU Compiler Collection auf Windows Systeme und ermöglicht dem Programmierer C, C++ und weitere Compiler unter Windows zu nutzen.

***Mock-up – Der Begriff Mock-up bezeichnet eine Art Entwurf bzw. einen einfachen Prototyp. Die nur zur Veranschaulichung von Benutzeroberflächen, oder Präsentationszwecken dienen soll und später mit voller Funktionalität umgesetzt wird.***

**Parser – ein Parser ist ein Programm oder ein Teil eines Programmes das dazu entwickelt worden ist eine Eingabe nach einem festgelegten Schema zu durchsuchen und in ein für die Weiterverarbeitung brauchbares Format zu wandeln.**

**POS – Point of Sale**

***Zu Deutsch Verkaufsort, bezeichnet aus Sicht des Käufers die Einkaufsstelle und aus Sicht des Verkäufers die Verkaufsstelle.***

**Proof of Concept - *Anhand eines Prototypen wird die prinzipielle Durchführbarkeit eines Vorhabens***

***getestet und je nach Ausgang eine Entscheidung über die weitere Durchführung eines Projektes gefällt.***

***Proprietäre Software – ist ein Begriff für nicht Quelloffene Software. Meisten geht dieser Begriff mit Kommerziellen Absichten einher. Proprietäre Software ist aber nicht zwangsläufig Kostenpflichtig, sondern nur nicht Quelloffen.***

**Qt** - Ein sehr umfangreiches, starkes und gut dokumentiertes Framework für Programme mit vorwiegend grafischen Benutzer Schnittstellen. Wird derzeit noch von Nokia unterstützt, um Anwendungen der nächsten Generation auf Mobile Plattformen zu bringen. Darüber hinaus gibt es eine Vielzahl von Software, die mit Hilfe dieses Frameworks entwickelt wurden. Sowohl Wissenschaftliche als auch Militärische Einrichtungen verwenden Software auf dieser Basis. Dank dieses Frameworks kann heute selbst unter C++ noch effektiv und schnell programmiert werden.

***Scene graph - ist eine Datenstruktur, die häufig bei der Entwicklung computergrafischer Anwendungen eingesetzt wird. Der Begriff Szenengraph ist nur unscharf definiert. Dies liegt daran, dass konkrete Szenengraphen in der Regel anwendungsgetrieben entwickelt werden.***

**Semantic Navigation -** Diese Technik bieten manche modernen IDEs, um dem Entwickler Vorschläge zu machen, auch welche Methode oder Member variablen er in einem bestimmten Semantischen Kontext zugreifen kann. Diese Funktion zeigt dem Entwickler schnell, ob er möglicherweise in einem beliebigen Umfeld noch keinen Zugriff auf bestimmte Methode oder Variablen hat. Dies spart zusätzlich Zeit, da der Entwickler nicht erst beim Kompilieren den Fehler entdeckt.

**Syntax Highlighting** - Ist eine Technik in modernen Editoren, die es ermöglicht Sprachspezifische Elemente Farblich hervor zu heben, um so dem Programmierer einen schnelleren Blick auf bestimmte Sprachelemente zu ermöglichen.

**Telemetrie – auch Fernmessung bezeichnet ein Verfahren, bei dem Daten an einem Ort aufgezeichnet werden und zu einer räumlich getrennten Sammelstelle übertragen werden. Die Daten können in der Sammelstelle entweder archiviert oder ausgewertet werden.**

**User-Interface – Oberbegriff für Benutzerschnittstellen zwischen dem Anwender und dem System. Die Aufgabe eines User-Interfaces ist es dem Benutzer seine Aufgabe so leicht wie möglich zu gestalten. Dabei spielen viele Faktoren eine Rolle wenn es um die Akzeptanz einer Software geht.**

**Vending – ist ein Begriff der im weiteren Sinne Verkaufsautomaten bzw. Selbstbedienungsautomaten bezeichnet.**

**Widget** - Als Widgets werden Komponenten grafischer Oberflächen bezeichnet, die den Benutzer die eigentlichen Schnittstellen zur Interaktion mit dem Programm dienen (Bsp. Eingabezeilen, Buttons, Textfelder etc.).

1. 6 Quellangaben

- <http://de.wikipedia.org/wiki/Mockup>

- <http://en.wikipedia.org/wiki/Scene_graph>

1. 7 Anlageverzeichnis

Anlage A - Pflichtenheft