Frage zu den Dokument-Namen: so wie Anleitung oder wie auf Kriterienliste? Unterteilung frei?

### Milestones

* SE Model festgelegt (Scrum, UP, ..)
* Projektplan inkl Plan für Benutzerbeobachtung (OK des Kunden) und Aufgabenteilung
* Vision Dokument V0 verabschiedet mit Preliminary Personas & Scenarios
* Benutzerbeobachtung/Befragung beendet
* Visionsdokument V1 (mit Personas und Key Szenarios) (formelles OK Stolze, inhaltliches OK Auftraggeber)
* Super-Set der "User Stories" (vollständiges Backlog, Sammlung von möglichen Anwendungs-Features mit Klassifizierung als Must/Should/Could) entwickelt (formelles OK Stolze, inhaltliches OK Auftraggeber)
* Domain Model (formelles OK Stolze (UML), inhaltliches OK Auftraggeber)
* Iteration 1 Feature Set (oder User stories) festgelegt (alle Features; Meeting mit Auftraggeber; Review mit Stolze)
* Abschluss von Iterationen
* Review der geplanten Aufwände der User Stories und Tasks für die nächste Iteration
* Architekturprototyp Demo, alle Komponenten & Designentscheide begründet (mündlich)
* Review Inhaltsverzeichnis Bericht
* Review Draft Wiki Page und Video

### Dokumente

* Aufwand dokumentiert (nachgeführte Arbeitsliste pro Person) und sinnvoll analysiert (PieCart)
* Es wird eine Liste mit offenen Tasks (mit Zuständigkeit) und abgeschlossenen Tasks (mit "Done Date") geführt
* Es wird eine Liste mit Risiken (und aktuellen Impediments) geführt (zugreifbar) und Änderungen kommuniziert
* Sitzungsprotokolle sind vollständig mit: Author, Datum der Erstellung, Teilnehmer, Beschlüsse, Datum des OK
* Reviews dokumentiert
* Tests dokumentiert (Was, wann, wer, welche Resultate)

### Kriterien Lösung Generell (Bereich "Inhalt")

* Nützlichkeit und Kreativität der Lösung
* Komplexität des Technologieumfeldes
* Vollständigkeit der Lösung (dokumentierte Abdeckung der funktionalen Anforderungen)
* Einhaltung der Nicht-Funktionalen Anforderungen (dokumentiert)
* Angemessenheit des externen Designs

### Kriterien für die Berichte

* Sprache (Rechtschreibung /Grammatik)
* Sprachfluss / Lesbarkeit / Verständlichkeit
* Keine Generalisierungen und Allgemeinplätze (alle Aussagen sind nachvollziehbar; Beispiele)
* Konsistenz des Sprachgebrauchs (Abkürzungsverzeichnis, Glossar)
* Abbildungen nummeriert und beschriftet und im Text referenziert
* Sinnvolle und einheitliche Strukturierung der Dokumente
* Quellen korrekt zitiert (gängigen Zitierstil eingehalten; WebLinks mit Seitentitel und "last visited")
* Wikipedia Quellen mit Permalink zitiert
* Mindestens ein Buch, ein ACM und IEEE Paper zitiert
* **Abstract / Management Summary Dokument**
* Abgenommen mit wenigen Iterationen / Änderungen?
* Problem (und warum ist das Problem ein Problem: Problemkontext, Vorgeschichte, was ist das Ziel)
* Systemarchitektur, Features, Vorgehen ("die Lösung")
* Was wurde erreicht (welche Ziele erreicht, welche zum Teil, zusätzliche "Benefits"; deutlich beschrieben welche Teile nicht bzw. unvollständig implementiert wurde, welche nicht getestet wurden).
* Lessons Learned bzgl. eingesetzter Methoden, Technologie etc.  
  inkl. Aufwandanalyse (Grafiken) mit Aufteilung nach interessanten Arbeitskategorien
* Abschluss-Satz
* **Vision Dokument**
* Abgenommen mit wenigen Iterationen / Änderungen
* Beschreibt mindestens ein glaubhaftes Benutzerproblem
* Enthält initiale Stakeholderanalyse (wer kauft die SW; wer braucht die SW; indirekte Nutzer)
* Enthält eine nützliche Auflistung der "Konkurrenz" aus Sicht Nutzer (inkl. "buy nothing")  
  Konkurrenzanalyse kann auch separates Dokument sein
* Enthält eine erste Beschreibung eines minimalen nützlichen Systems (was muss V1 können für die Pilotnutzer)
* Dokument wird nicht nachgeführt (Freeze nach "Project Go")
* **Personas, Szenarios, User Analyse Dokument**
* Abgenommen mit wenigen Iterationen / Änderungen
* Personas sind glaubwürdig und haben Referenz auf Interview log; gewählte Persona(s) sind fokussiert
* Szenarios sind glaubwürdige "Future" Szearios und haben Referenzen auf Interview log
* Optional (wo sinnvoll) Storyboards
* Optional (wo sinnvoll) Ist Szenations
* Interviews sinnvoll dokumentiert (Transcript nicht notwendig)
* **Anforderungen/Software Requirements Dokument**
* Abgenommen mit wenigen Iterationen / Änderungen
* Alle User Stories (oder Use Cases "brief") dokumentiert (mit Datum: identified, started, tested, completed)
* Alle aus User Stories ableitbaren NF Anforderungen dokumentiert
* (optinal) Alle aus User Stories ableitbaren Features dokumentiert
* (optional) Feature / User Story Matrix zeigt Herkunft von Features
* Alle Design Constraints dokumentiert
* Diskussion ob Accessability notwendig enthalten
* **Domain Analyse Dokument (Daten, GUI)**
* Abgenommen mit wenigen Iterationen / Änderungen
* Keine Design Elemente
* Inhaltlich korrekt; komplett; aktuell (entspricht Code)
* Domain Model ist korrektes UML
* UI angepasst an Persona: Können, Ziele, Situationen
* User Environment Diagram oder Screen Map für Anwendungen mit mehreren Screens
* Design-Entscheide und Redesignentscheide dokumentiert
* Dokumentation von frühen Papier-Prototypen
* Einhaltung GUI Guidelines und Standards (dokumentiert welche genutzt wurden)
* Diskussion der Eignung für Farbenblinde und andere Accessibility Guidelines (wenn angebracht)
* Dokumentiert, dass UI für Iteration alle notwendigen User Stories bzw System Features abdeckt
* **Architektur Dokument (Entwurf)**
* Abgenommen mit wenigen Iterationen / Änderungen
* Angemessenheit der Architektur (sind alle Design Entscheide dokumentiert und aus User Stories begründet)
* Saubere Schichtenarchitektur (incl MVC/MVVM; loose coupling, gute Package-Struktur):   
  Ideal: Beweis der Schichtung (zB. NDepend); Beschreibung der Layer Interfaces
* Korrekte Anwendung von Patterns
* Sinnvolle Verwendung von UML zur Dokumentation (z.B. Zustandsdiagram)
* Alle wichtigen Arch Decisions aufgelistet (Technlogiewahl sauber dokumentiert)
* Alle Arch-Decisions / System Eigenschaften sind auf Anforderungen (oder Constraints) zurückgeführt (keine unbegründeten Arch-Features); Arch Decision mit Tests und Evaluationen substanziert
* Wichtige Elemente der Architektur diskutiert und beschrieben.
* Matrix von App-Features und supporting Arch-Features. (Welches Feature der Applikation wurde mit welcher architektonischen Lösungsvariante realisiert)
* **Test Dokument**
* Testabdeckung (Unit Tests) dokumentiert und sinnvoll
* Manuelle Tests gut beschrieben, mit Log wann wer was mit welchem Erfolg getestet hat.
* Abdeckung der Use-Cases  (bzw. User Stories / Features) durch Tests dokumentiert
* Tests decken in Gesammtheit alle funktionalen und nichtfunktionalen Anforderungen ab
* System-Test gut dokumentiert
* Usability Tests gut dokumentiert  (Aufgaben dokumentiert und gut)
* Tests wurden zum Ende der Iterationen durchgeführt und dokumentiert (Test Datum)
* **Code Dokument**
* Einfaches importieren in Entwicklungsumgebung beschrieben und möglich (local Dev)
* Unit Tests laufen in der abgegebenen Lösung auf Prof. PC, oder/und Screenshot von erfolgreichen Tests
* Files mit Header information; Author; Datum von Code Reviews
* Separate JavaDoc oder entsprechendes; Generiert aus Code; Dokumentiert wichtige APIs
* Code sinnvoll dokumentiert
* Code Qualität inklusive Beweis der Qualität mit Metric Tools (Ndepend, Resharper, PMD, Checkstyle, QJ-Pro, Jdepend, Hammurapi, Jlint, DoctorJ, … (http://www.java2s.com/Product/Java/Byte-Source-Code/Source-Analysis-Diagram.htm)
* Code stimmt mit Architektur überein (idealerweise dokumentiert durch Analyse Tool)
* Coding Standards sind dokumentiert und werden eingehalten (idealerweise dokumeniert durch Analyse Tool)
* Keine Codeleichen (ungenutzte Klassen & Interfaces); Kein auskommentierter Code
* Sinnvolle Verwendung von Patterns
* Keine "bad smells"
* Sinnvolle Nutzung von Libraries
* Platformspezifische Coding Patterns (z.B. Activties in Android und Thumbstoning) werden eingehalten
* Keine unnötigen Warning etc (am besten dokumentiert mitttels Screen Shot)
* **Benutzer/Installationsdokumentation**
* Unterschiedliche Dokumentation für unterschiedliche Rollen (zB. Admin, User)
* Enthält ein Kapitel "Erste Schritte" welches den Nutzer durch eine erste sinnvolle Nutzung führt
* Dokumentation orientiert sich an wichtigen Nutzungsszenarien und zeigt mindestens ein vollständiges "Sunny Day" Szenario.
* Optional: Wenn sinnvoll Bereitstellung einer Quick-Card/Cheat-Sheet mit einer Kurzanleitung

### Weiteres

* Einsatz von Tools sinnvoll & dokumentiert (SVN, Track, UnitTest, CodeQuality, …)
* Arbeit gemäss Plan
* Gutes Mgmt des Auftraggebers: Erwartungen werden gemanaged: keine Überraschungen
* Selbsständigkeit & Einsatz (zB Kommunikation mit Auftraggeber)

### Weitere Deliverables

* Video (YouTube)
* Projekt Extended Management Summary & Wiki Page (Anzahl Reviews)
* SA "Brochure Text" (möglichst wenig Reviews)
* Erfahrungsbericht für jedes Teammitglied + Lessons Learned
* Poster (attraktiv, kein Fliesstext)

### Com (Email)

* Sitzungsprotokolle und Change msgs nicht mehr als 1 Arbeitstag nach Durchführung (pub auf Wiki)
* E-Mail be Zeitüberscheitungen um +/-30% zu plan pro Woche
* Vor jeder Sitzung wird Agenda geschickt (z.B. Content auf Wiki; eMail= Link)

### Abgabe

* Termingerecht (bis 12 Uhr)
* Wikiseite
* YoutTube Video (verlinkt auf Wikiseite)
* Installation/Ausführung der Software laut Anleitung möglich
* Abgabe aller Teile der Arbeit auf CD (mit leicht navigierbarem Inhaltsverzeichnis: z.B. html)
* Abgabe des Berichtes in einem PDF mit Inhaltsverzeichnis und durchgehenden Kapitel- und Seitennummern
* Abgabe der unterschriebenen Erklärungen, Vereinbarungen und Aufgabenstellung als Teil des Pdf