

Qualitätssicherungsplan

SE-Gruppe 207

Version 0.9

Dokument-Historie

Version	Status	Datum	Verantwortlicher	Änderungsgrund
0.1	created	13.04.2004	LaBr	Creation
0.5	changed	18.04.2004	GeHa	Hinzufügen QS-Methoden, Lay-outänderungen
0.6	changed	18.04.2004	GeHa	Hinzufügen Zeitplan
0.9	changed	18.04.2004	LaBr	Überarbeitung Kap. 1-3

Inhaltsverzeichnis

Inhaltsverzeichnis	2
1. Einleitung	4
1.1. Zweck des Dokuments	4
1.2. Gültigkeit des Dokuments	4
1.3. Begriffsbestimmungen und Abkürzungen	4
1.4. Zusammenhang mit anderen Dokumenten	4
2. Projektübersicht	5
2.1. Projektbezeichnung	5
2.2. Projektorganisation	5
2.3. Projektkurzbeschreibung	5
3. QS - Anforderungen	5
3.1. Beginn	6
3.1.1. Projektplan	6
3.1.2. Visionsdokument	6
3.1.3. Anwendungsfalldiagramm	6
3.1.4. Versionsmanagement	6
3.1.5. UI Prototyp	6
3.1.6. Softwarearchitektur	7
3.1.7. Domänenmodell	7
3.1.8. Nichtfunktionale Anforderungen	7
3.1.9. Anwendungsfallbeschreibung	7
3.1.10. Projektrisiken	7
3.1.11. Programmierrichtlinien	7
3.1.12. Dokumentationsrichtlinien	7
3.2. Ausarbeitung	7
3.2.1. UI – Prototyp	8
3.2.2. Executable Prototyp	8
3.2.3. Testplan/Testfälle	8
3.3. Konstruktion	8
3.4. Übergang	8
3.4.1. Installationsleitfaden	9
3.4.2. Benutzerhandbuch	9
3.4.3. Projektendbericht	9
3.4.4. Projektarchiv	9
4. Allgemeine QS-Methoden	10
4.1. Reviews	10
4.1.1. Dokumentenreviews	10
4.1.2. Statusreviews	10
4.1.3. Kunden- bzw. Anwenderreview	10
4.1.4. Software-Inspektion	10
4.2. Audits	11
4.3. Tests	11
4.4. sonstige QS-Methoden	11
4.4.1. Software Engineering	11
4.4.2. Dokumentation	11

4.4.3. Richtlinien	12
5. Produkte.....	12
5.1. Wie werden die Produkte qualitätsgesichert:.....	12
6. Zeitplan.....	13
7. Verantwortlichkeit	13

1. Einleitung

1.1. Zweck des Dokuments

Der Zweck des vorliegenden Dokuments ist es, alle geplanten qualitätssichernden Maßnahmen für das Projekt Ticketline zu dokumentieren. Er dient damit als Leitdokument für alle Projektmitarbeiter bei der Durchführung und dem Nachweis aller QS-bezogenen Tätigkeiten (Reviews, Tests, Korrektur- und Vorbeugungsmaßnahmen, Q-Berichtswesen, ...).

1.2. Gültigkeit des Dokuments

Dieses Dokument ist gültig für die Entwicklung vom Projekt Ticketline. Der Plan findet Anwendung in den Phasen

- Initiierung
- Spezifikation
- Entwurf
- Implementierung
- Integration
- Test

Der Plan ist gültig für die gesamte Projektdauer.

1.3. Begriffsbestimmungen und Abkürzungen

TL:

Ticketline

1.4. Zusammenhang mit anderen Dokumenten

Dieser Plan regelt die Maßnahmen zur Qualitätssicherung von folgenden Produkten:

- Projektplan
- Visionsdokument
- Versionsmanagement
- Softwarearchitektur
- Domänenmodell
- Nichtfunktionale Anforderungen
- Anwendungsfallbeschreibung
- Projektrisiken
- UI – Prototyp
- Executable Prototyp
- Testplan/Testfälle
- Programmierrichtlinien
- Installationsleitfaden
- Benutzerhandbuch
- Projektendbericht
- Projektarchiv
- Dokumentationsrichtlinien
- Begriffsverzeichnis
- Geschäftsregeln

- Meetingprotokolle
- Meilensteintrendanalyse
- Technische Infrastruktur

2. Projektübersicht

Dieses Kapitel bietet eine kurzgefasste Übersicht über das Projekt Ticketline.

2.1. Projektbezeichnung

Als Bezeichnung für dieses Projekt wurde der Name „Ticketline“ gewählt.

2.2. Projektorganisation

Projektleiter:

Als Projektleiter dient bei dieser Übung Josef Trojer.

QS - Verantwortliche:

Die Teilnehmer der Übungsgruppe: Martin Marcher, Gerald Haider, Lachlan Brazier

Programmierer:

Derzeit noch nicht bekannt. Unser Tutor dient als Kontaktperson zur Implementierungsgruppe.

Tester:

Laut Übungsangabe muss jedes Übungsmitglied auch Whitebox Tests erstellen.

Auftraggeber:

RISE Institut, TU Wien

Ansprechpartner des Auftraggebers:

Die Kontaktperson ist Josef Trojer

2.3. Projektkurzbeschreibung

Das Produkt Ticketline soll für die Informationsgewinnung, Verwaltung und Verkauf von Karten (eng. Tickets) für alle Art von Veranstaltungen dienen.

Der Auftraggeber möchte sein bisheriges System zum Kartenverkauf auf ein Computersystem umstellen, um für geplante, zukünftige Expansionspläne gerüstet zu sein.

Weitere Informationen findet man im Dokument <Visionsdokument>.

3. QS - Anforderungen

In diesem Kapitel werden alle Produkte und Prozesse aufgeführt, die qualitätsgesichert werden.

Auf Grund der Anforderung des Auftraggebers wird in diesem Dokument nur der Entwicklungsprozess qualitätsgesichert. Dazu wird der Entwicklungsprozess laut Projektplan in folgende Phasen aufgeteilt:

- Beginn
- Ausarbeitung
- Konstruktion

- Übergang

Die Phasen Wartung und Auslaufen des Produktes werden hier nicht qualitätsgesichert.

Die Produkte werden auf Vollständigkeit, Korrektheit, Konsistenz, Änderbarkeit und Verständlichkeit geprüft. Zur Überprüfung der Produkte dienen Inspektionen.

3.1. Beginn

In der Phase Beginn sind laut Projektplan folgende Dokumente gefordert:

- Projektplan
- Visionsdokument
- Anwendungsfalldiagramm
- Versionsmanagement
- UI Prototyp
- Softwarearchitektur
- Domänenmodell
- Nichtfunktionale Anforderungen
- Anwendungsfallbeschreibung
- Projektrisiken
- Programmierrichtlinien
- Dokumentationsrichtlinien

3.1.1. Projektplan

Der Projektplan soll die einzelnen Phasen und die erforderlichen Produkte des Projekts beschreiben.

Die Gesamtprojektdauer, und die Dauer der einzelnen Phasen sollten festgehalten werden. Des weiteren sollen die Projektkosten abgeschätzt werden.

3.1.2. Visionsdokument

Im Visionsdokument soll das Produkt beschrieben werden. Warum wurde das Produkt gefordert, was kann es leisten?

3.1.3. Anwendungsfalldiagramm

Dieses Dokument soll alle Anwendungsfälle, Aktoren und deren Kommunikation untereinander beschreiben.

3.1.4. Versionsmanagement

Hier soll beschrieben sein, wie Versionen zu benennen sind, welche Versionen von welchen Produkten erstellt werden sollen und wie die Versionen zu archivieren sind.

3.1.5. UI Prototyp

Hier soll eine erste Version des User Interfaces erstellt werden. Die Funktionalität muss noch nicht voll entwickelt sein. Es fördert aber der Qualität, wenn zumindest viel benutzte UI Interaktionen schon machbar sind, auch wenn die Funktionalität dahinter noch nicht implementiert ist.

3.1.6. Softwarearchitektur

Hier soll beschrieben werden, welche Software Komponenten zum System gehören, und wie diese interagieren.

Welche Software Komponenten werden wiederverwendet?

Welche Vorgaben an Robustheit, Verlässlichkeit und Sicherheit gibt es?

Ist die Architektur soweit unterteilt, das die Entwicklung von mehreren Personen gemacht werden kann?

Sind alle externen Schnittstellen beschrieben?

3.1.7. Domänenmodell

In diesem Dokument sollen die Objekte, deren Attribute und Methoden beschrieben sein. Bei Datenbankanbindungen sollen hier auch die EER Diagramme stehen.

Können die Objekte die Anwendungsfallbeschreibungen abdecken?

3.1.8. Nichtfunktionale Anforderungen

Hier sind jene Anforderungen beschrieben, die nicht die Funktionalität beschreiben, jedoch Einfluss auf das Produkt Ticketline haben. Beispiele wären Qualitätsanforderungen, Leistungsanforderungen, Fehlerverhalten etc.

3.1.9. Anwendungsfallbeschreibung

In diesem Dokument werden die Anwendungsfälle beschrieben. Jeder Anwendungsfall muss folgende Punkte enthalten:

- Eindeutige Nummer
- Titel
- Kurzbeschreibung
- Vorbedingungen
- Genauer Ablauf der Ereignisse. Jedes Ereignis erhält eine eindeutige Nummer. Jede Antwort wird ebenfalls gekennzeichnet.
- Auswirkungen auf Systemzustand und Datenbestand
- Anmerkungen

3.1.10. Projektrisiken

In diesem Dokument werden die Projektrisiken aufgelistet. Unerwartete Ereignisse bzw. Verzögerungen in den einzelnen Phase sind typische Vertreter von Risiken.

3.1.11. Programmierrichtlinien

Hier soll beschrieben werden, wie der Code auszusehen hat. Namenskonventionen, Einrückungen, etc. sollten beschrieben sein.

3.1.12. Dokumentationsrichtlinien

In diesem Dokument soll die Struktur aller Dokumente beschrieben sein. Ein wichtiges Augenmerk soll die Versionsbehandlung sein.

3.2. Ausarbeitung

In der Phase Implementierung sind laut Projektplan folgende Produkte gefordert:

- UI – Prototyp

- Executable Prototyp
- Testplan/Testfälle

3.2.1. UI – Prototyp

Es gilt das gleiche wie in Kapitel 3.1.5.

Es sollen aber auch geänderte Anforderungen überprüft werden.

3.2.2. Executable Prototyp

Die erste Lauffähige Version soll betrachtet werden. Dazu macht man auch eine Code Inspektion des bisher geschriebenen Codes.

Sind die Programmierrichtlinien eingehalten worden? Gibt es logische Fehler? Macht das System was es schon machen soll?

3.2.3. Testplan/Testfälle

Der Testplan gibt Auskunft über das Testvorgehen und der Testmethode.

Zumindest die Tests für die wichtigsten Anwendungsfallbeschreibungen, und deren Objekte sollen beschrieben sein.

Er soll enthalten:

- Teststrategie
- Testgegenstände und ihre zu testenden Eigenschaften (Testfälle)
- Testzeitplan
- Testmethoden
- Kriterien für Gelingen/Misslingen eines Testlaufes
- Kriterien für den Start der Testdurchführung
- Kriterien für eine Unterbrechung des Tests
- Testendprodukte
- Kommunikationswege zwischen Tester und Produktverantwortliche
- Art der Tests
- Konfiguration der Testplattform

3.3. Konstruktion

In dieser Phase soll das System erstellt werden. Es soll darauf geachtet werden, dass der entwickelte Code mit den Vorgaben und dem Design zusammenpasst.

3.4. Übergang

In der Phase Implementierung sind laut Projektplan folgende Produkte gefordert:

- Installationsleitfaden
- Benutzerhandbuch
- Projektendbericht
- Projektarchiv

3.4.1. Installationsleitfaden

Hier soll beschrieben sein, wie das System zu installieren ist. Vorbedingungen, Konfigurationsanweisungen, etc. müssen beschrieben sein.

3.4.2. Benutzerhandbuch

Hier soll beschrieben werden, wie das System zu bedienen ist. Betrachtet werden die Sichten eines Administrator und eines Endbenutzers.

3.4.3. Projektendbericht

Hier werden noch einmal alle Ergebnisse des Projekts zusammengefasst.

3.4.4. Projektarchiv

In diesem Dokument steht, wie und wo die Produkte des Projekts archiviert werden.

4. Allgemeine QS-Methoden

Allgemein kann und soll man folgende Methoden zur Qualitätssicherung (von Softwareprojekten) einsetzen.

4.1. Reviews

Reviews sind eine der Kernmöglichkeiten zum sicheren von Qualität. Reviews werden primär zum Sicherstellen der Qualität von Dokumentation verwendet.

Die Grundidee hinter Reviews ist, ein gewisses Dokument/Projektergebnis von externen Experten beurteilen bzw. überarbeiten zu lassen. Hierbei ist das extern besonders hervorzuheben, da z.B.: der Verfasser eines Dokumentes, dieser jenes sehr schwer wirklich kritisch beurteilen kann.

Die übliche Abhaltungsweise eines Reviews ist das persönliche Zusammentreffen aller am Reviewprozess beteiligten Personen.

Diese Personen lassen sich in die folgenden Rollen einteilen:

- Moderator
- Autor (wenn nicht QS-intern)
- Reviewer
- Protokollschreiber

4.1.1. Dokumentenreviews

Dokumentenreviews überprüfen den Inhalt von Dokumenten bzgl. Vollständigkeit, Konsistenz und Korrektheit. Weiters wird kontrolliert, ob die Dokumentationsrichtlinien für die Erstellung des Dokuments eingehalten worden sind.

4.1.2. Statusreviews

Bei einem Statusreview werden Soll- und Ist-Zustand eines Projektes/Dokuments verglichen und bei ungewünschten Abweichungen werden Gegenmaßnahmen geplant.

4.1.3. Kunden- bzw. Anwenderreview

Das Produkt wird durch den Kunden- bzw. den Anwender auf Erfüllung seiner Anforderungen überprüft.

4.1.4. Software-Inspektion

Hierbei geht es um die Identifikation und Definition von möglichen Defekten in Softwareelementen.

Es werden dabei Sourcedateien, Konfigurationsdateien, Skripte, ... gegen die Spezifikation und gegen die Richtlinien auf Vollständigkeit, Konsistenz und Korrektheit überprüft.

4.2. Audits

Grundsätzlich liegt das Abhalten eines Audits sehr nahe bei der Abhaltung eines Reviews, dennoch gibt es einige sehr wichtige Unterschiede.

Zum einen ist der Kleine aber sehr wesentliche Unterschied, dass bei einem Audit projekt-fremde Personen miteinbezogen werden. Dadurch eröffnet sich für alle Teilnehmer oft eine ganz andere Sicht auf das Produkt.

4.3. Tests

Tests sind einer der QS Methoden im Bereich von Softwareprodukten.

- Funktionstests
 - Black-Box
 - White-Box
 - Zustandstests
- Systemtests
 - Leistungstests
 - Usability Test
 - Security Test

Die Funktionsweise der Tests soll hier nicht näher erläutert werden, da dies in anderen Lehrveranstaltungen viel genauer behandelt wird.

4.4. sonstige QS-Methoden

Hier noch etliche, nicht ganz unwichtige, Maßnahmen zum Qualitätssichern von Softwareprojekten.

4.4.1. Software Engineering

Durch den Einsatz einer standardisierten und erprobten Software Engineering-Methode wird die Projektdurchführung nach einem allgemein definierten und erprobten Schema durchgeführt. Als mögliche Methoden kämen hier der „Rational Unified Process“ oder das „Microsoft Souldution Framework“ in Frage. Die Vorgehensweisen und Prozesse der Softwareentwicklung werden dadurch nachvollziehbar und transparent. Außerdem können durch Anwendung eines solchen Prozesse, viele Fehler schon im Vorhinein vermieden werden.

4.4.2. Dokumentation

Nur durch eine entsprechende Dokumentation während der Projektausführung sind die Abläufe, Entscheidungen nachvollziehbar. Anschließende Wartungsarbeiten können beim Vorhandensein von entsprechenden Dokumenten schneller durchgeführt werden. Mündlich ausgetauschte Informationen werden zumeist durch das Stille Post-Prinzip verfälscht, teilweise überhaupt vergessen oder später anders dargestellt. Alle Informationen, welche nicht schriftlich vorliegen, verlieren schnell an Wert. Eine in entsprechender Qualität und Quantität vorliegende Dokumentation ist eine maßgebliche Voraussetzung für eine erfolgreiche Projektausführung und um Qualitätssicherung betreiben zu können.

4.4.3. Richtlinien

Mittels der Richtlinien werden die Prozesse im Projekt standardisiert und es wird eine Einheitlichkeit der produzierten Komponenten sichergestellt. Die verschiedenen Richtlinien müssen jeweils zu Projektbeginn global definiert werden und anschließend natürlich von allen beteiligten Personen bis ins Detail befolgt werden.

Im der Regel sollten die folgenden Richtlinien definiert werden:

- Dokumentationsrichtlinien
- Programmierrichtlinien
- Versionierungsrichtlinien
- Sourceverwaltung

5. Produkte

5.1. Wie werden die Produkte qualitätsgesichert:

Dokument/Produkt	QS-Methode
Projektplan	Dokumentenreview
Visionsdokument	Dokumentenreview
Versionsmanagement	Dokumentenreview
Softwarearchitektur	Dokumentenreview
Domänenmodell	Dokumentenreview
Nichtfunktionale Anforderungen	Dokumentenreview
Anwendungsfallbeschreibung	Dokumentenreview, Kundenreview
Projektrisiken	Dokumentenreview
UI-Prototyp	Anwenderreview, Kundenreview
Executable Prototyp	Anwenderreview, Kundenreview
Tesplan/Testfälle	Dokumentenreview
Programmierrichtlinien	Dokumentenreview
Installationsleitfaden	Dokumentenreview
Benutzerhandbuch	Dokumentenreview, Anwenderreview
Projektendbericht	Dokumentenreview
Projektarchiv	Dokumentenreview
Dokumentationsrichtlinien	Dokumentenreview
Begriffsverzeichnis	Dokumentenreview
Geschäftsregeln	Dokumentenreview
Meetingprotokolle	Dokumentenreview
Meilensteintrendanalyse	Dokumentenreview, Statusreview
Technische Infrastruktur	Dokumentenreview, Kundenreview

6. Zeitplan

untenstehend eine Auflistung der geplanten QS-Maßnahmen:

Zeitraum KW	Dokument/Produkt	Öfters/einmal
	Projektplan	Öfters
	Anwendungsfalldiagramm	Öfters
	Visionsdokument	Einmal nach ersten Prototyp
17	Versionsmanagement	Einmal
17	UI-Prototyp	Wird erst ab 2. Prototyp Re- viewed da möglicherweise der Projektauftraggeber diesen vollkommen umgestaltet
17	Softwarearchitektur	Öfters
17	Domänenmodell	Öfters
17	Nichtfunktionale Anforderungen	Öfters
17	Anwendungsfallbeschreibung	Öfters
18	Projektrisiken	Einmal
21	UI-Prototyp	Einmal
21	Executable Prototyp	Einmal (Funktionstest)
21	Tesplan/Testfälle	Öfters
21	Programmierrichtlinien	Einmal
26	Installationsleitfaden	Öfters
26	Benutzerhandbuch	Öfters
26	Definition was ist eine "Version 1.0"	Einmal
27	Projektendbericht	Einmal
27	Projektarchiv	Einma

7. Verantwortlichkeit

In diesem Kapitel werden die Verantwortlichkeiten für die einzelnen Produkte festgelegt. Da versucht wird, jedem Übungsteilnehmer die gleiche Arbeit zu geben, können sich diese Verantwortlichkeiten noch ändern.

Dokument/Produkt	Verantwortlicher
Projektplan	Lachlan Brazier
Visionsdokument	Lachlan Brazier
Versionsmanagement	Martin Marcher
Softwarearchitektur	Lachlan Brazier
Domänenmodell	Martin Marcher
Nichtfunktionale Anforderungen	Gerald Haider
Anwendungsfallbeschreibung	Lachlan Brazier
Projektrisiken	Martin Marcher
UI-Prototyp	Gerald Haider
Executable Prototyp	Alle
Testplan/Testfälle	Lachlan Brazier
Programmierrichtlinien	Gerald Haider
Installationsleitfaden	Martin Marcher
Benutzerhandbuch	Alle

Projektendbericht	Alle
Projektarchiv	Gerald Haider
Dokumentationsrichtlinien	Lachlan Brazier
Begriffsverzeichnis	Lachlan Brazier
Geschäftsregeln	Martin Marcher
Meetingprotokolle	Martin Marcher
Meilensteintrendanalyse	Martin Marcher
Technische Infrastruktur	Martin Marcher
