Pflichtenheft

FennecBot

Autor: Team FennecBot
Letzte Änderung: 12. Mai 2021
Dateiname: Pflichtenheft_FennecBot.docx

Version: 0.3

Inhaltsverzeichnis

Copyright

Version Historie

Version:	*Datum:*	*Verantwortlich*	*Änderung*
0.1	04.05.2021	Lukas Evers	Initiale Dokumenterstellung
0.2	10.05.2021	Son Khue Nguyen	Erweiterungen
0.3	11.05.2021	Umut Uzunoglu	Erweiterungen
0.4	12.05.2021	Hien Anh Ngyuen Manh	Erweiterungen
0.5			
1.0			
1.1			
1.2			

1 Vorhandene Dokumente

Alle für die vorliegende Spezifikation ergänzenden Unterlagen müssen hier aufgeführt werden

Dokument	Autor	Datum
Lastenheft	Team FennecBot	12.05.2021
Technische Spez. JetRacer	Waveshare	

2 Überblick

Hier soll eine kurze Zusammenfassung des zu erstellenden Moduls erfolgen

Nur ein grober Überblick der wesentlichen Punkte

Quasi ein Abstract vom ganzen Pflichtenheft

Für das Projekt soll auf Basis eines Waveshare JetRacers ein autonom fahrendes Auto erstellt werden. Die dazu zu erstellenden Teile bestehen aus einem 3D - gedruckten Chassis und der Software zur Verarbeitung der Sensordaten, um die Motoren des JetRacers entsprechend anzusteuern. Die Software wird auf einem Nvidia Jetson Nano ausgeführt. Die Software soll mittels Bilderkennung einen kleinen Fußball erkennen und dann den JetRacer autonom zum Ball fahren lassen um ihn zu schießen oder ihn zu dribbeln. Neben dem autonomen Modus kann man das Auto auch manuell mit einem Controller steuern.

Hauptziele

Tabellarische Darstellung der Ziele basierend auf dem Lastenheft

#	Ziel	Beschreibung der Implementation
1	Erkennung eines Balles	mit ROS Package über Intel Realsense (Tiefenkamera)
2	Erkennung einer Linie auf dem Boden	find_2d/3d_object ROS Package über Intel Realsense
3	Autonomes Verfolgen und Abfahren einer Linie	Computer Vision zusammen mit ROS Controller für die Ansteuerung der Motoren
4	Autonomes Erkunden und Suche nach einem Ball falls nicht im Sichtfeld des Roboters	ROS Navigation mit Intel Realsense und LIDAR für SLAM-Funktionalität
5	Steuerung FennecBot mit einem Remote Controller	ROS Teleop zusammen mit ROS Controller
6	Autonomes Vermeiden von Hindernissen welches nicht Bälle sind	ROS Navigation mit Tiefenkamera und LIDAR
7	Erkennung eines Tores für den Ball	find_2d/3d_object ROS Package über Intel Realsense
8	Beförderung des Balles in das richtige Tor	ROS Navigation, Intel Realsense, LIDAR
9	Variierung der Fahrgeschwindigkeit für Anpassung an verschiedene Situation (viele, wenig Hindernisse, Erkundung, Schießen, Dribblen)	Phasenerkennen und Übermittlung der Fahrgeschwindigkeit an den ROS Controller (Steuerung über PWM)
10	Rückwärtsfahren und autonom rangieren	ROS Controller und ROS Navigation

#	Ziel	Beschreibung der Implementation
1	Erkennung eines Balles	Computer Vision
2	Erkennung einer Linie auf dem Boden	Computer Vision
3	Autonomes Verfolgen und Abfahren einer Linie	Computer Vision zusammen mit ROS Controller für die Ansteuerung der Motoren
4	Autonomes Erkunden und Suche nach einem Ball falls nicht im Sichtfeld des Roboters	ROS Navigation mit Tiefenkamera und LIDAR für SLAM-Funktionalität
5	Steuerung FennecBot mit einem Remote Controller	ROS Teleop zusammen mit ROS Controller
6	Autonomes Vermeiden von Hindernissen welches nicht Bälle sind	ROS Navigation mit Tiefenkamera und LIDAR
7	Erkennung eines Tores für den Ball	Computer Vision
8	Beförderung des Balles in das richtige Tor	Computer Vision, ROS Navigation, Tiefenkamera, LIDAR
9	Variierung der Fahrgeschwindigkeit für Anpassung an verschiedene Situation (viele, wenig Hindernisse, Erkundung, Schießen, Dribblen)	Phasenerkennen und Übermittlung der Fahrgeschwindigkeit an den ROS Controller (Steuerung über PWM)
10	Rückwärtsfahren und autonom rangieren	ROS Controller und ROS Navigation

3 Annahmen und Abgrenzungen

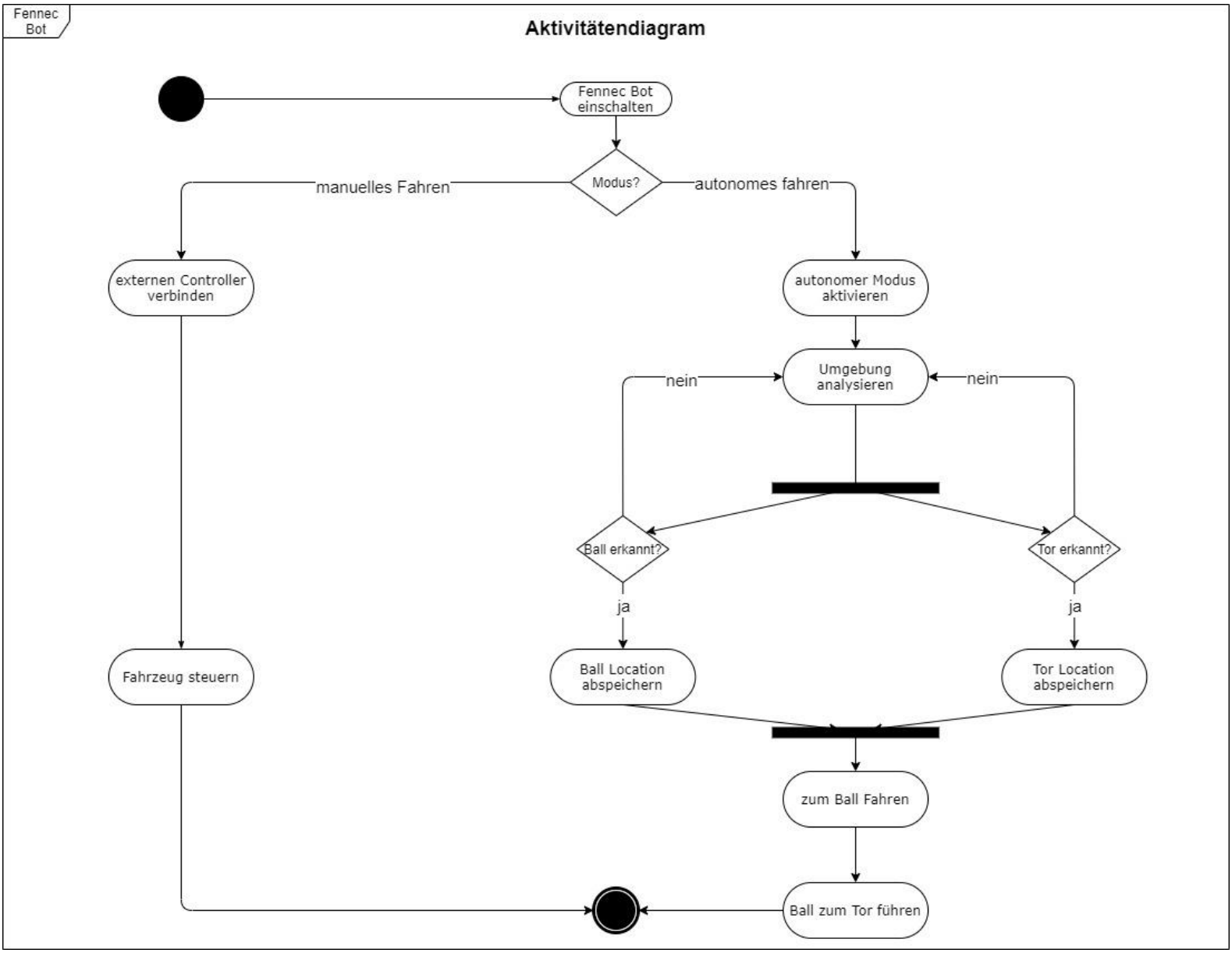
#	**Annahmen (fachliche und technische Annahmen)**
1	Verwendung von Open-Source-Code ist ohne Einschränkungen möglich
2	Verwendung eines Waveshare JetRacer
3	Einsatz von ROS
4	Bilderkennung über Intel Realsense
5	
6	

#	**Abgrenzungen (Was ist in dieser Lösung nicht enthalten bzw. abgedeckt)**
1	
2	
3	
4	
5	
6	

4 Workflow

Graphische Einbindung der Workflows

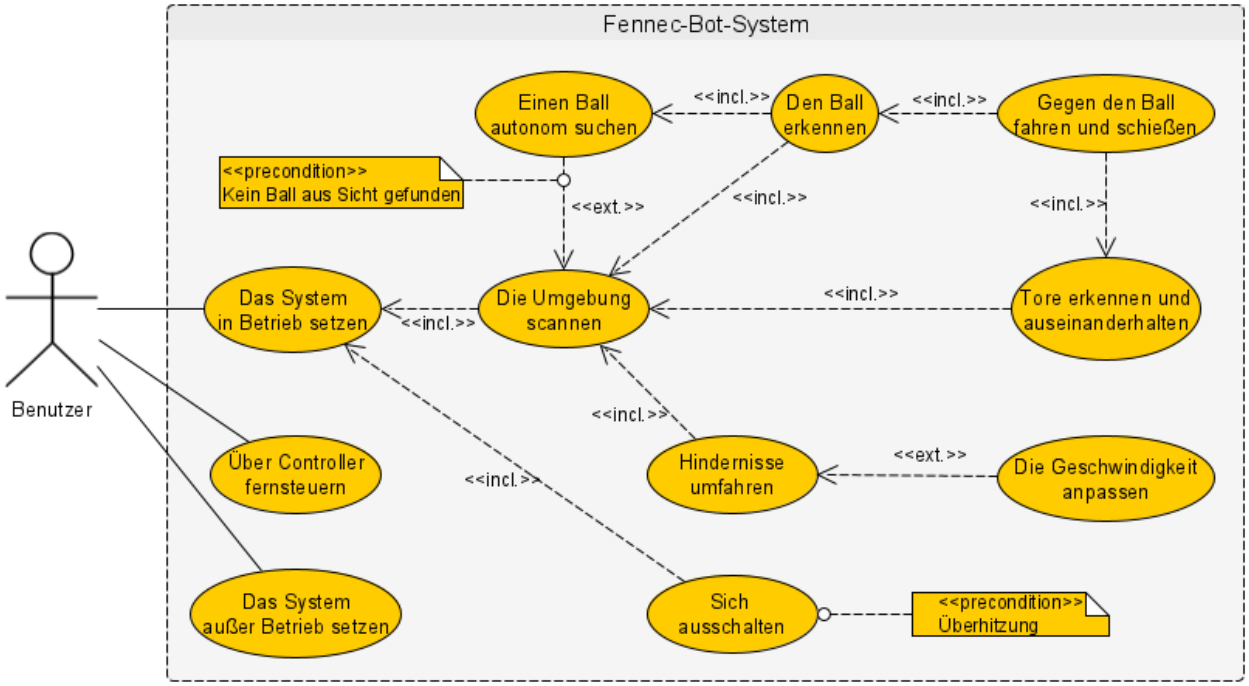
Aktivitätendiagramm? Ablaufdiagramm?



5 Funktionalität

5.1 Überblick

Auflistung der Hauptfunktionen, die im folgenden Abschnitt jeweils detailliert erläutert werden unter Bezugnahme zu den zuvor definierten Workflows



5.2 Funktion 1: Die Umgebung scannen

Zweck/Ziel	Der Fennec kann eine Karte über die Intel Realsense/Laserscanner (LiDAR) von der Umgebung erzeugen, Hindernisse umfahren, einen Ball autonom anfahren und suchen/erkennen
Akteur/Auslöser	LiDAR/Intel Realsense
Vorbedingung	keine
Daten-Input	Tiefendaten/-bilder, Farbbilder, Laserscandaten
Verarbeitungsschritte	1. Intel Realsense/LiDAR nehmen die Umgebung auf 2. Durch SLAM werden Laserscandaten in ROS visualisiert
Ergebnis	Der Fennec ist sich seiner Umgebung bewusst und kann dementsprechend eine Entscheidung zur nächsten Bewegung treffen: z. B. Ball suchen, anfahren, Hindernisse umgehen
Plausibilitäten	Zu überprüfen: ● Die Reichweite und die Aktualisierungsgeschwindigkeit der gescannten Karte, wenn sich der Fennec mit unterschiedlicher Tempo bewegt ● Die Reaktionsschnelligkeit bei der Begegnung mit verschiedenen Objekten (mehr dazu in Plausibilitätsprüfungen Funktionen 2, 4, 6)
Fehlerhandling	● Kein Ball gefunden? → Weiter suchen! ● Die Sicht wird komplett blockiert → Still stehen und warten
Anforderung	*Verweis auf Anforderungsdokument, -liste, etc.* *Verlinkung auf Lastenheft hier einfügen*
Test Cases	● Erkennen und eingrenzen des Spielfeldes ● Hindernisse, die die Fahrt beeinflussen umgehen ● Den Ball ins Tor führen

5.3 Funktion 2: Einen Ball autonom suchen und erkennen

Zweck/Ziel	Der Fennec kann einen Ball autonom suchen und erkennen, macht somit sein Ziel und die Fahrtrichtung fest und fährt in Richtung Ball
Akteur/Auslöser	Intel Realsense, Motoren, Achse, ROS Control
Vorbedingung	Funktion 1 muss erfüllt sein, damit die Suche eingeschränkt werden kann
Daten-Input	Tiefendaten/-bilder, Farbbilder
Verarbeitungsschritte	1. Durch ROS Packages wird ein Ball als Zielobjekt antrainiert. Wobei die Bilder mit der Intel Realsense gemacht werden 2. Das Package sucht dann mit der Intel Realsense den Ball im Raum und gibt entsprechend den Abstand zum Ball über Tiefendaten zurück 3. Über ROS Navigation wird der Fennec zum Ball bewegt, falls er gefunden wurde. ROS Control übergibt dem Fennec (Motoren und Achse) die benötigten Parameter, um den Ball zu erreichen 4. Wurde er nicht gefunden, soll der Fennec sich im Spielfeld/Raum bewegen und den Ball suchen
Ergebnis	Der Ball wurde gefunden und der Fennec fährt auf den Ball zu. Darauf folgt das Führen/Schießen des Balles
Plausibilitäten	Zu überprüfen: ● Die maximale und minimale Entfernung, in der der Ball erkannt werden kann ● Überlastungsmöglichkeit bei Sichtung von mehr als einer Ball
Fehlerhandling	● Ball wird nicht erkannt → mehr Bilder als Input für die Erkennung ● Fennec bewegt sich zu schnell und erkennt den Ball nicht → Mehr FPS für die Bilderkennung (max. 60 FPS) FPS = Frames per Second
Anforderung	Verweis auf Anforderungsdokument, -liste, etc. Verlinkung auf Lastenheft hier einfügen
Test Cases	● Rollenden Ball erkennen ● Stillen Ball erkennen

5.4 Funktion 3 Gegen den Ball fahren und schießen

Zweck/Ziel	Der Fennec kann mit maximaler Geschwindigkeit gegen den Ball fahren und schießen, um einen Treffer zu erzielen/den Ball vorzulegen und hinterher zu fahren.
Akteur/Auslöser	Motoren, Achse, ROS Control
Vorbedingung	Funktion 2 wurde erfüllt und der Ball erkannt.
Daten-Input	• optional: Tele-OP (Controller Input) zur Steuerung
Verarbeitungsschritte	1. Nach der Erkennung des Balles fährt wird die Geschwindigkeit erhöht und die Achse in Richtung des Balles durch ROS Control bewegt (Geschwindigkeits- und Achsenkontrolle ist asynchron, deshalb nur ein Schritt). 2. Der Ball wird nicht als Hindernis erkannt und der Fennec fährt gegen den Ball mit voller Geschwindigkeit 3. Bei Tele-OP Steuerung wird der Fennec manuell bedient
Ergebnis	Der Ball wurde gefunden und der Fennec fährt auf den Ball zu. Darauf folgt das Führen/Schießen des Balles
Plausibilitäten	Zu überprüfen: • Die größte Kraft, die auf den Ball ausgeübt werden kann, ohne dass sich der Fennec selbst schadet • Die geringste Kraft, die auf den Ball ausgeübt werden kann, um ihn zu bewegen
Fehlerhandling	• Der Ball wird aufgrund des Geländes oder des Winkels der gerichteten Kraft in eine unerwartete Richtung bewegt → •
Anforderung	Verweis auf Anforderungsdokument, -liste, etc. Verlinkung auf Lastenheft hier einfügen
Test Cases	• Gegen stillliegenen Ball fahren und schießen. • Gegen leicht rollenden Ball in einer gewissen Entfernung fahren und schießen.

5.5 Funktion 4: Hindernisse umfahren

Zweck/Ziel	Der Fennec kann sowohl Hindernisse als auch Geländeschwierigkeiten erkennen und seine Route entsprechend anpassen.
Akteur/Auslöser	Motoren, Achse, ROS Control
Vorbedingung	keine
Daten-Input	Tiefendaten
Verarbeitungsschritte	1. Nach der Erkennung Hindernisse verlangsamt der Fennec seine Geschwindigkeit 2. Eine optimalere Route wird berechnet und der Fennec dreht seine Achse entsprechend 3. Der Fennec folgt der neuen Route und kehrt zur normalen Geschwindigkeit zurück
Ergebnis	Die Fennec meiden die Hindernisse und beeinträchtigen die originale Route so wenig wie möglich.
Plausibilitäten	Zu überprüfen: ● Die Zeit, die der Fennec benötigt, um sich von schwer zu navigierenden Hindernissen umzuleiten, z.B. weitläufige Straßensperre ● Die Größe und Position der toten Winkel bei der Bewegung mit dem Ball
Fehlerhandling	● Der Fennec steht vor einer Sackgasse → Bei der Berechnung der neuen Route muss auch die Rückverfolgung berücksichtigt werden ● Einige Hindernisse werden vom Ball blockiert → Bessere Vermeidung von toten Winkeln
Anforderung	Verweis auf Anforderungsdokument, -liste, etc. Verlinkung auf Lastenheft hier einfügen
Test Cases	● Hindernisse unterschiedlicher Größe werden in unterschiedliche Richtungen und Winkel gesetzt. ● Plötzlich auftauchte Hindernisse

5.6 Funktion 5: Automatisch abschalten

Zweck/Ziel	Der Fennec kann langsamer werden und sich selbst abschalten, nachdem bestimmte Bedingungen erfüllt sind.
Akteur/Auslöser	Motoren, CPU
Vorbedingung	Die Innentemperatur erreicht eine bestimmte Schwelle.
Daten-Input	Temperatur
Verarbeitungsschritte	1. Die CPU wird durch lange Nutzung aufgeheizt, bis zu einer vorbestimmten Temperatur 2. Der Fennec wird langsamer 3. Der Fennec schaltet sich automatisch ab
Ergebnis	Der Fennec ist ausgeschaltet und muss vor dem erneuten Einschalten abgekühlt werden.
Plausibilitäten	Zu überprüfen: Die höchste Temperatur, die die CPU aushalten kann
Fehlerhandling	
Anforderung	● optional: Verweis auf Anforderungsdokument, -liste, etc.
Test Cases	Die Abgrenzung der CPU testen

5.7 Funktion 6: Tore erkennen

Zweck/Ziel	Der Fennec soll wissen, was er mit dem Ball machen soll, heißt er muss die Tore als Ziel definieren können, damit er den Ball auch ins Tor führen/schießen kann
Akteur/Auslöser	find_2d/3d_object package, Intel Realsense
Vorbedingung	keine Vorbedingung
Daten-Input	Farb- und Tiefenbilder
Verarbeitungsschritte	1. Die Intel Realsense ist immer aktiv und wird neben dem Ball auch für Tore zur Bilderkennung antrainiert. 2. Das ROS Package gibt dann wichtige Daten (wie z.B. Koordinaten im Raum) zurück
Ergebnis	Der Fennec kann mit den gegebenen Informationen den Ball ins Tor führen/schießen
Plausibilitäten	Zu überprüfen: Die Entfernung, aus der der Fennec ein Tor unterscheiden kann
Fehlerhandling	● Der Tor ist blockiert → ● Die Kennzeichnung auf dem Tor wird getauscht →
Anforderung	● optional: Verweis auf Anforderungsdokument, -liste, etc.
Test Cases	● Fennec vor dem Tor mit/ohne Ball platzieren und das Tor erkennen lassen ● Fennec während Bewegung mit/ohne Ball das Tor erkennen lassen ● Aus verschiedenen Perspektiven das Tor erkennen lassen

Zweck/Ziel	Ziel dieser Funktion Was soll sie erledigen
Akteur/Auslöser	User, Rolle, System, Batch-Prozess etc.
Berechtigung	Welche Rechte sind notwendig, um diese Funktion auszuführen?
WF-Rererenz	Referenz zum zugehörigen Workflow mit Angabe z.B. Step-Nummer o.ä.
Vorbedingung	● z.B. Bildstatus = in Bearbeitung ● z.B. Arbeitsschritt = zugewiesen ● z.B. Berechtigung ● ….
Daten-Input	● Welche Daten sind als Input für dieser Funktion notwendig? ● Was muss an Daten und in welchem Status vorliegen, damit dieser Funktion gestartet wird ● ggf. Verweis auf notwendige Schnittstellen, falls diese vorgesehen sind
Verarbeitungsschritte	Beschreibung der Verarbeitungslogik gemäß Workflow Definition. Hier erfolgt die eigentliche Beschreibung dieser Funktion 3. Schritt 1 4. Schritt 2 5. etc.
Ergebnis	Was steht als Ergebnis nach Ausführung dieser Aktion? In welchen Status befinden sich wichtiger Attribute?
Plausibilitäten	Auflistung mögliche Pluasi's, Attribute-stati, B
Fehlerhandling	Was passiert im Fehlerfall? z.B. ● bei falscher Eingabe erneute Aufforderung zur Eingabe richtiger Daten ● Daten unvollständig in diesem Step, Meldung Daten vervollständigen o.ä. ● …..
Folgeprozess	ggf. Verweis auf die direkt folgenden Prozesse bzw. Funktionen im Workflow
Out of Scope	Was ist in dieser Funktion nicht enthalten und out of scope zu sehen ist!
Anforderung	● optional: Verweis auf Anforderungsdokument, -liste, etc.
Release	Für welchen Release ist dieser Funktion vorgesehen? z.B. 1.8
Test Cases	● Optionale Auflistung möglicher Testszenarien für diese Funktion

Anschließend können Screen-Shots (GUI) zur Verdeutlichung der Funktion angezeigt werden
Sonstige Erläuterungen, die notwendig sind, um den Sachverhalt zu verdeutlichen sind ebenfalls hier zu ergänzen
z.B. Haken-Konzept der verschiedenen Schalter für LWM o.ä. etc.

6 Offene Fragen

#	Issue	Status	Owner	Deadline
1	Offene Fragen an Auftraggeber.	O (offen) B (in Bearbeitung) A (abgeschlossen)	Müller	datum

7 Modul Abhängigkeiten

- Welche Abhängigkeiten gibt es zu anderen Modulen
- Einfache Aufzählungsform