

**PSS-Bericht**



|  |  |
| --- | --- |
| **Projektname** | Definition und Automatisierung von Equipment Test Cases für ein IMA System |
| **Firma** | Konzept Informationssysteme GmbH, Meersburg |
| **Zeitraum** | 01.09.2018 – 28.02.2019 |
| **Name** | Jeehun Chang |
| **Studiengang, Matrikel Nr.** | AIT, 293423 |
| **Betreuer der Firma** | Dipl.-Ing. Jochen Zwick |
| **Gutachter der Hochschule** |  |

Inhaltsverzeichnis

[1 Vorwort 4](#_Toc1948996)

[2 Projektbeschreibung 5](#_Toc1948997)

[3 Projektumgebung 6](#_Toc1948998)

[3.1 GitLab 6](#_Toc1948999)

[3.2 Testanlage 7](#_Toc1949000)

[3.3 VMware 7](#_Toc1949001)

[3.4 CoToRDC 7](#_Toc1949002)

[3.5 Python 8](#_Toc1949003)

[3.6 Pycharm 8](#_Toc1949004)

[3.7 Daedalos 8](#_Toc1949005)

[3.8 Doors 8](#_Toc1949006)

[4 Arbeitspakete 9](#_Toc1949007)

[4.1 A429 to A664 / A664 to A429 9](#_Toc1949008)

[4.2 AI / AO 9](#_Toc1949009)

[4.3 DSO / DSI 9](#_Toc1949010)

[4.4 Gateway 9](#_Toc1949011)

[5 Projektdurchführung 10](#_Toc1949012)

[5.1 Konfiguration 10](#_Toc1949013)

[5.2 Testimplementierung & Modifizierung 10](#_Toc1949014)

[5.3 Dokumentierung & Review 10](#_Toc1949015)

[6 Fazit 11](#_Toc1949016)

[7 Quellen 12](#_Toc1949017)

Abbildungsverzeichnis

# Vorwort



Abbildung 1 Konzept Informatinossystem GmbH

Konzept Informationssysteme ist ein Software- und Systemhaus für Industrieunternehmen mit den Schwerpunkten Softwareentwicklung, System Engineering und Qualitätssicherung im süddeutschen Raum, welches 20 Jahres Erfahrung in der Umsetzung von anspruchsvollen Projekten im IT-Umfeld hat. Dabei ist Konzept nicht nur auf eine Branche spezialisiert, sondern deckt ein Leistungsspektrum von Analyse und Projektierung über Softwareentwicklung und Qualitätssicherung bis hin zu Beratung und Schulung ab. Geründet wurde die Firma 1994 und hat in diesem Zeitraum eine Größe von ca. 140 Mitarbeitern erreicht, welche sich auf die Standorte Meersburg, Ulm, München und Hünenberg(CH) aufteilt.

Die Auftraggeber kommen aus den unterschiedlichsten Branchen wie zum Beispiel Avionik, Automotive, Raumfahrt, Energiesysteme, Produktion und Logistik sowie Verteidigungstechnik, Bahntechnik und Medizintechnik.

# Projektbeschreibung

Das Projekt geht es um die Überprüfung eines neuen IMA (Integrierte Modular Avionics) System auf die Einhaltung von definierten Anforderungen.

IMA bezeichnet einen distributiven Echtzeitcomputer an Bord eines Luftfahrzeugnies, mit standardisierten Komponenten und Schnittstellen zur Kommunikation zwischen den unterschiedlichen Systemen.

Es wurde im Vorfeld an das Projekt ein ERD (Equipment Requirements Document) formuliert, in welchem alle System Anforderungen enthalten sind. In dem Dokument wird ebenfalls zwischen den verschiedenen Typen von Anforderungen unterschieden. Die Anforderungen sind aufgeteilt in Analysis, Performance und Funktional. Eine Art der Verifikation der jeweiligen Anforderungen ist die Formulierung von Test Cases, in welchen die Verhaltensweise der Testanlage nach Anforderungen verifiziert wird. Mit Hilfe von Test Cases lassen sich Funktionale und Performance Anforderungen prüfen.

Zunächst sollen über einen Zeitraum von sechs Monaten Anforderungen der CAN, A429, AI/AO und DSI/DSO überprüft werden. In den Anforderungen sind Übertragung von Daten Typen, sowie Zustände und Signale beschrieben, in denen sich die Testanlage befinden, beziehungsweise senden soll, wenn bestimmte Fälle eintreten. Anhand dieses Dokumentes werden vom Team passende Test Prozeduren und Test Cases formuliert und ausgeführt, in denen geprüft wird ob die vorher gegebenen Anforderungen erfüllt werden.



Das Projekt wurde Anfang Juli im Jahr 2018 begonnen. Für das neue Produkt gibt es zurzeit noch keinen Hard- und Software Prototypen, deshalb überprüft die Anforderungen mit Hilfe eines älteren CRDC Systems. Die beiden Systeme

# Projektumgebung

## GitLab

GitLab ist eine webbasierte Anwendung für Softwareprojekte um die Versionsverwalutng von Dateien zu verteilen. Es bietet nicht nur die Funktion als Speicher an, sondern auch mehrere nützliche Funktionen für Entwickler.

* Repository

Repository ist einen Platz zur Anwendung der Speicherung und Mitteilung aller Dateien, was für Projekts benötigt. Es ist auch möglich auf dem Repository direkt die Dateien zu korrigieren. Das Repository ist für jeden Branch separat und kann über Berichtigungen zugegriffen werden.

* Branches

Branch ist der große Vorteil von Git und die meisten Versionverwaltungssysteme bieten die Brunches. Ein Branch ist eine Kopie des gesamten Codes, der gerade entwickelt wird, und die weitere Entwicklung wird unabhängig vom ursprünglichen Code ausgeführt.

Das Hauptbranch wird Master branch genannt und es wird automatisch generiert bei der Herstellung von neues Git. Nach der Anlegung eines neuen Branchs aus Master Branch arbeiten die Entwickler auf ihrem lokalen Branch und die Änderungen werden danach auf den Master zurück gemerged. Da das lokale Branch vor dem Merge durch CI überprüft wird, ob das Bruanch fehlerlos und vollständig ist, ist es realisierbar, mögliche Fehler durch die Änderung gar nicht erst in den Master Branch zu kommen.

* Issues

Issue ist eine Einheit der Arbeit, dazu gehören zum Beispiel Entwicklung der Funktionen, Bug-Fix usw. Es enthält auch zusätzliche Informationen über die Arbeit, zu wem zugewiesen wurde, wie hohe Priorität hat und in welchem Zustand sich befindet. Gitlab bietet auch eine Issue Tracking Funktion, die eine Histrorie aller Änderungen hinterlässt. Der Status von dem Issue ist nur offen und geschlossen, so dass es nicht möglich ist, ein Issue, das nicht rechtzeitig abgeschlossen wurde, zu verkleien oder zu verbergen. Ein Issue wird in der Reihenfolg Registrierung, Zuweisung, Fortschritt, Überprüfung und Abschluss bearbeitet.

* CI / CD

Wenn die Branches für jeden Entwickler mehrere erstellt werden, ändert sich jedes Branch, und alle Codes werden in ihre Richtungen aufgeteilt. Wenn diese Branches für einen bestimmten Zeitraum verwendet werden, steigt die Schwierigkeit der Integration, die Zeit für die Konfliktlösung nimmt zu und das Fehlerrisiko auch steigt. Um diese Situation zu überwinden, haben wir eine CI (kontinuierliche Integration) eingeführt. Durch die CI können kleine Änderungen im Entwicklungsprozess beibehalten und die Version des Repositorys durch Überprüfung des Codes in regelmäßigen Abständen kontrolliert werden. Entwicklungsteams, die eine CI-Plattform verwenden, beginnen häufig mit Einstellungen für die Versionskontrolle und Ausführungsdefinitionen.

Der Entwickler bearbeitet den Code in seinem Editor in der lokalen Umgebung. Wenn der Entwickler die Änderungen push oder merge, wird Der neu kombinierte Code an eine CI-Pipeline gesendet. Die CI-Plattform führt dann die erforderlichen Code-Analyse-Tools aus und stellt den Bericht den Entwicklern zur Verfügung, wenn Probleme mit dem Build auftreten. Fehlerraten werden aufgezeichnet, und diese Dateien können verwendet werden, um Verbesserungen im Entwicklungsprozess zu erkennen. Sobald das Build und Verify erfolgreich abgeschlossen ist, gibt das CI-Plattform das Code frie und diese Stufe als CD (Kontinuierliche Delivery) bezeichnet.

* Wiki

Innerhalb der Projektumgebung auf Gitlab ist eine Dokumetationsfunktion implementiert, in der berechtigte Anwender wichtige Informaitonen oder Prozess des Projektes dokumentieren können. Dies kann zum Beispeil als Einsteiger Anleitung dienen, sodass falls ein neuer Entwickler dem Projekt beitritt, dieser alle wichitgen Informationen zur Einrichtung direkt verfügbar hat.

* Charts

In Gitlab ist es dem Anwender möglich das Projekt mithilfe von gesammelten Daten und generierten Graphen zu analysieren. Zum einen ist es möglich das Projekt auf Commites zu überprüfen, diese werden als Kreisdiagramm und in zeitlicher Abfolge als Balkendiagramm. Zum anderen ist es möglich die ausgeführten Pipeline Jobs zu analysieren, hier wird zusätzlich zu den zeitlich angeordneten Graphen auch die Passed zu Failed Rate angezeigt, welche aufzeigt wie sorgfältig im Projekt anhand von den Anforderungen gearbeitet wurde. Eine Teaminterne Zielvorgabe war es bei den Commits eine Passed zu Failed Rate von etwa 80% zu erreichen, dies würde dafür sorgen, dass das Projekt von Beginn an sauber geführt wird.

## Testanlagen

Die Testanlagen, die auf dem Projekt verwendet werden, sind beim Kunde in den Testschränken montiert, die SATE (S Automatisierung Test Equipment) oder Bench genannt. Eine SATE ist mit zwei Rechnern verbunden, die SATE PC genannt, einen um die Testanlage zu steuern und einen um Data zu laden.

Über die virtuellen Maschinen können wir uns zusätzlich über eine weitere Remoteverbindung auf den SATE PC anmelden. Zum einen befindet sich hier der Daedalos Server, zum anderen lassen sich hier mit ADS-2 Session Manager neue Konfigurationen laden oder es werden weitere Debugging Anwendungen bereitgestellt.

Über den SATE PC können die Stromversorgung der Testanlagen bedienen, so dass einzelne oder die gesamte Testanlage ein oder ausschalten. Da mehrere Messgeräte in die Testanlage integriert sind, ist es möglich vor Ort nahezu alle wichtigen Daten zu messen.

## VMware

Beim Kunden ist die Testanlagen vorhanden, dabei handelt es sich um einen Schrank, in welchem diverse Testsysteme verbaut sind. Daneben uns auch Mitarbeiter des Kunden bei Bedarf auf der Testanlage arbeiten, befindet sich die Testanlage nicht in der Firma Konzept. An der Testanlage ist ein Computer angeschlossen, über welchen die Testanlage gesteuert wird. Dieser PC ist mit Hilfe von virtuellen Maschinen, welche wir zur Verfügung gestellt bekommen von der Ferne aus zu erreichen. Der Zugang auf de VM erfolgt über den Horizon Client von VMware, mit Hilfe eines RSA Security Tokens.

## CoToRDC

CoToRDC ist ein Konfiguration Tool für CRDC zur Anwendung der Generierung des Media-Sets. Das Media-Set enthält die Informationen der Übermittlungen der Dateien aus einem Input zu einem Output von CRDC. Nach dem Media-Set an der Testanlage hochgeladen wird, können die bei der Testanlage konfigurierte Routings überprüft werden. Deshalb ist es notwendig mit dem CotoRDC ein neues Media-Set zu generieren, bevor ein Testskript zur Überprüfung des neuen Routings laufen zu lassen.

Alle Konfiguration des Media-Sets können an dem CoToRDC manuell erstellt werden und die können durch die Ladung der AC-ICD (Aircraft Interface Control Ducument) leichter generiert werden. AC-ICD ist ein Dokument, das die Interface- Konfigurationsparameter enthält. Mit Hilfe der AC-ICD ist es möglich, den Aufwand der Generierung zu reduzieren und über die konfigurierten Routings übersichtlich zu schauen und zu dokumentieren.

## ADS2

ADS2 ist

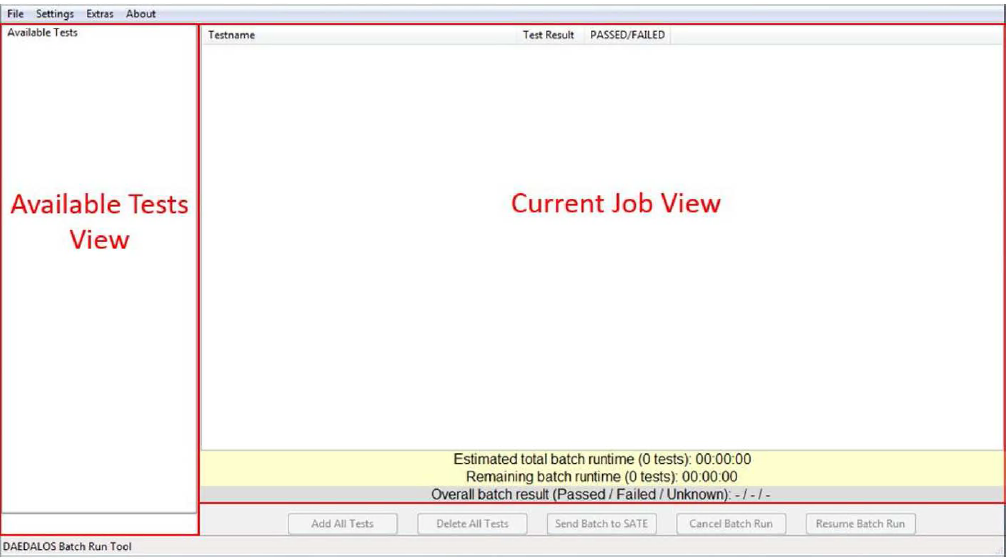
## Python

Python ist

## PyCharm

Pycharm ist

## Daedalos



Daedalos ist eine Client-Server basierte kundenspezifische Anwendung zur Organisation von Testskripten. Auf dem SATE PC befindet sich das Tool als Server und auf der virtuellen Maschine befindet sich das Tool als Client. Durch Deadalos-Client werden eine oder mehrere verfügbare Tests zum SATE PC gesendet, werden die Inhalten des Testskripts zu Deadalos-Server kopiert. Dann lässt Deadalos-Server den kopierten Test auf der Testanlage laufen. Während des Testlaufens informiert das Tool den Job Name, den Username, die Remaining Duration und die aktuelle Test Status. Nach der Durchführung des Tests werden die Resultate von Server zu Client zurückgegeben. Dann kommen die Ergebnisse auf dem Current Job View mit den übersichtlichen Farben (Rot: Failed, Grün: Passed, Gelb: Unkown)

Zwar über mehrere VMware ist es möglich gleichzeitig Testskripts an die Testanlage zu übermitteln, aber auf der Testanlage wird der Test einzeln ausgeführt. Daedalos bietet die Funktion zur Zeiterstellung an, dadurch ist es möglich, die langlaufende Tests vor oder nach der Arbeitszeit laufen zu lassen, um die lange Wartezeit zu vermeiden.

## Doors

Doors ist

# Arbeitspakete

Konzept Informationssystem hat für die Bearbeitung des Auftrages mehrere Arbeitspakete zugewiesen, von denen vier während meiner Praxissemester abgedeckt werden.

## A429 to A664 / A664 to A429

Auf das Arbeitspaket werden die Routings A429 (ARINC 429) zu A664 (ARINC 664) und A664 zu A429 überprüft.

* A429

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 bit | 2 bits | 19 bits | 2 bits | 8 bits |
| Parity bit | SSM | Payload | SDI | Label |

Inhalt

Versiedene code typ. BNR\_INTEGER, float, BCD-integer,float, (ssm länge unterschiedlich), boolean, bitfield, opaque, string, hybrid bnr integer, float, bcd integer, float.

Je nach der ssm werden datentyp und negetive zahlen entschieden.

Es gibt 12 unterschiedliche ssm,

* A664

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 bytes | 1 byte | 6 bytes | 6 bytes | 2 bytes | 46 bytes | | | | | 4 bytes | 12 bytes |
| Preamble | Start Frame  Delimiter | Destination Addresse | Source Addresse | 0x800 Ipv4 | IP Structure  20 bytes | UDP Structure 8 bytes | AFDX Payload  20 bytes | Padding 0 to 16 | SN 1 byte | Frame Check Seq | Inter Frame Gap |
|  |  |  |  | | | |  |  | | |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  | 1 byte | 1 byte | 1 byte | 1 byte | 4 byte | 4 byte | | 4 byte | 4 byte |  |
|  |  |  | FS\_01 | FS\_02 | FS\_03 | FS\_04 | Payload\_01 | Payload\_02 | | Payload\_03 | Payload\_04 |  |

Inhalt

## AI / AO

Über Arbeit Auf das Arbeitspaket werden alle Readbacks und Anweisungen überprüft, ob sie richtig gemessen werden bzw. richtig konfiguriert werden.

* AI\_-10\_+10VDC

Verify voltage rb

AI hat insgesamt 9 interfaces.

Load anhängen und switsch zu

Mit DAC (Digital Analog Cenverter) werte einsetzen, die auf dem AI Interface die Voltage fließen soll.

Prüfen ob konfigurierte Voltage wert kommen.

Durch scale factor ändert sich die Voltage werte auch.

+/- 12V damage.

* AO\_0\_+10VDC

AO hat 5 Interfaces.

Load anhängen,

Verify AO switch state, voltage rb, current rb, passivation, short circuit

Cmd: voltage, switch, reactivation

Internal problem: passivation switch off, short circuit, reactivation

möglich AO voltage cmd und AO switch cmd default wert

loc prüfen

## DSO / DSI

Über Arbeit alle function works richtig

* DSI

GND/OPN, CIF\_28V, CIF\_GND

Switch state, cmd

Verify Logic state polarity positive, negative

GND/OPN: OPN 0, Gnd 1 (Positive)

CIF\_28V: Opn 0, 28V 1 (POSitive)

CIF\_GND: Opn 0, GND 1 (POSITIVE)

* DSO

GND\_OPN\_250mA; 28V\_OPN\_250mA; CIF\_GND\_1A, CIF\_28V\_1A

Cif\_gnd, sw\_opn: volt 28V curt 0A

CIF gnd, sw\_close: volt 0, curt 1A

CIF\_28V, sw\_opn: Volt 0V, curt 0A

CIF\_28V, sw\_close: volt 28, curt 1A

Overload, short circuit

Using Virtualle load and load wert niedriger and current höher, short circuit passieren dann passivation state verify , danach reactivation cmd

switch default möglich

loc prüfen

## Gateway

Bis jetzt alle nach a664. Aber ab jetzt alle mögliche routings überprüfen werden.

Discret , analog, numeric, passivation, shortcircuit, overcurrent, passivation.

Alle routings in eine aicicd und test script so comliziert. Viele acicd und viele konfig erstellung benötig.

# Projektdurchführung

## Konfiguration Stufe

## Testimplementierung & Modifizierung Stufe

## Dokumentierung & Review Stufe

# Fazit

# Quellen

GitLab Docs

PD10000005031\_000\_06\_UM\_coToRDC.pdf

Daedalos\_system\_requirements\_documentation.pdf

