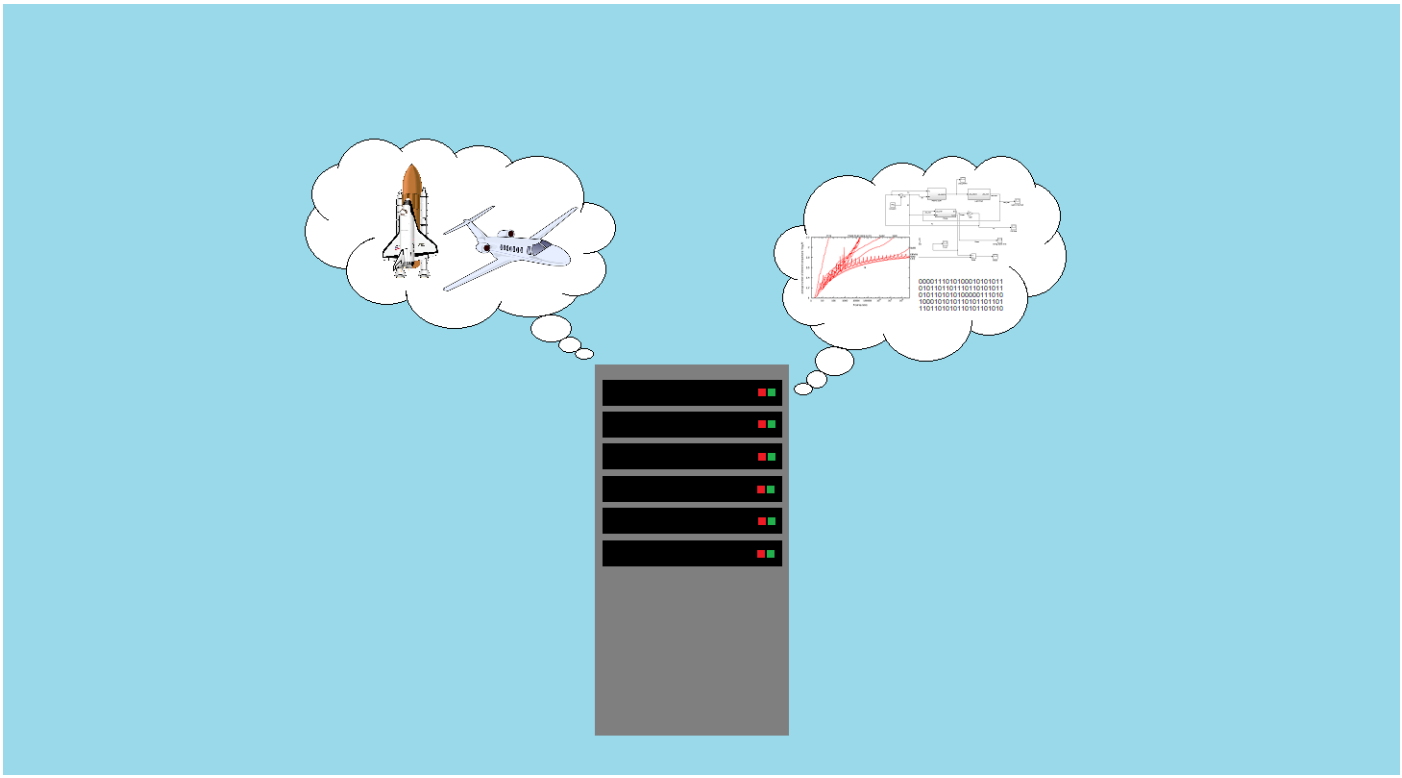
CHALMERS



Mjukvarutestning av Radar i en Simulerad Miljö

Examensarbete inom Data- och informationsteknik

JESPER HOLM

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

EXAMENSARBETE

Mjukvarutestning av Radar i en Simulerad Miljö

Examensarbete inom Data- och informationsteknik

JESPER HOLM

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2019

Mjukvarutestning av Radar i en Simulerad Miljö
Examensarbete inom högskoleingenjörsprogrammet Datateknik
JESPER HOLM

Examinator: Peter Lundin, Instutitionen för Data-och informationsteknik
Handledare: Erland Holmström, Instutitionen för Data-och informationsteknik,
Karin Thorvaldsson, SAAB AB, Fredrik Mårlind, SAAB AB

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Omslag:
”Testrigg som tänker”

Institutionen för Data- och Informationsteknik
Göteborg 2019

SAMMANFATTNING

Testsystemet för integration och verifiering av mjukvara till radarsystem är en trång resurs hos Saab. Det medför att testning av system tar lång tid att genomföra då stor del av testningen sker på fysisk hårdvara med begränsad tillgänglighet. Försvarsindustrin går mot mer digitala produkter och med det ökar konkurrensen i branschen.

Examensarbetet syftar till att undersöka hur man kan testa mjukvara i en simulerad miljö för att göra testningen oberoende av systemspecifik hårdvara och därmed mer lättillgänglig. Syftet med arbetet har uppnåtts genom att kombinera flera av Saabs existerande simulatorer för att bygga en testmiljö för mjukvara.

Arbetet fokuserar på radarsystemet som sitter på GlobalEye som är ett spaningsflygplan med avancerade sensorsystem. För att förstå hur man testar komplexa system så har en förstudie gjorts. Efter studien så har intervjuer med sakkunniga som arbetar nära testmiljön som används på Saab genomförts.

Resultat av arbetet är en implementation av en testmiljö som testar integrationen mellan signal- och databehandling mot radaroperatörernas användargränssnitt. Implementationen gjordes på kommersiell hårdvara utan några systemspecifika komponenter med hjälp av sakkunniga på Saab.

Nyckelord: radar, hårdvarusimulering, testning , Saab, signalbehandling, simulerad miljö, GlobalEye, C2

ABSTRACT

The test system for integration and verification of software for radar systems at Saab is a narrow resource. As a result, the testing becomes a time-consuming process, because the testing is done on real hardware. The digitalization of products are increasing and the concurrence is growing.

The aim of this thesis is to examine how testing of software can be done with a simulated environment. To make testing of software independent from system specific hardware and easy accessible. In addition, to look in to how and if it is possible to combine existing simulators that is used at Saab today, to build a test environment for software.

The system in focus is the radar system on GlobalEye, which is an airborne early warning and control aircraft with advanced sensor systems. To understand how testing of complex systems is done, a pre-study has been made. After the study was done, interviews with professionals working close to the current test environment were conducted.

The result is an implementation of a test environment that tests signal- and data processing integrated with the radar operators' user interface. The implementation was done on commercial hardware without any system specific components in collaboration with professionals at Saab.

Keywords**:** Saab,radar, simulated hardware, testing, simulated environment, GlobalEye, C2

FÖRORD

Mjukvarutestning av Radar i en Simulerad Miljö är ett examensarbete för högskoleingenjörer. Arbetet är utfört på dataingenjörsprogrammet vid institutionen för Data- och informationsteknik på Chalmers Tekniska Högskola. Arbetet är gjort i samarbete med Saab AB i Göteborg.

Jag skulle vilja tacka mina handledare Erland Holmström på Chalmers, Karin Thorvaldsson och Fredrik Mårlind på Saab för ett bra samarbete och handledning. Jag vill även tacka de personer som bistått med hjälp och visat engagemang under arbetets gång.

Jesper Holm, Göteborg, Februari 2019

INNEHÅLLSFÖRTECKNING

TERMINOLOGI

I&V – Integration & Verifiering
AEW&C – Airborne Early Warning & Control System, luftburet varning- och ledningssystem
Leveransriggen – Testmiljö på Saab bestående av skarp hårdvara
Referensriggen – Testmiljö på Saab bestående av både systemspecifik hårdvara och simulerade delar
GlobalEye - Komplext AEW&C system producerat av Saab
MTS – Mission Training System, system som används för utbildning av radaroperatörer
HWIL – Hardware In-The-Loop
HIL – Human In-The-Loop
SIL – Software In-The-Loop
MIL – Model In-The-Loop
MW – Middleware, mellanvara, mellanprogramvara
ECU - Electronic Control Unit
NASA – National Aeronautics and Space Administrations (USAs federala myndighet för luft- och rymdfart)
COTS – Commercial of the Shelf, kommersiella produkter
VMS - Virtual Memory System, minneshanteringsfunktion för temporärt skriva internminne till disklagring

1 INLEDNING

Bakgrund

Testsystemet för integration och verifiering av GlobalEye Airborne Early Warning & Control System är en trång resurs på Saab. Utvecklarna av delsystem för GlobalEye vill leverera system oftare samt få snabbare återmatning från en större del av systemet. Avdelningen för integration och verifiering fastnar lätt i ett ekorrhjul av uppstartsintegration. De har svårt att få tid till verklig verifiering och analys utan att stänga leveransriggen för inleveranser av mjukvara under några veckor. Försvarsindustrin går mot mer digitala produkter och med det ökar konkurrensen i branschen. På grund av detta så måste utvecklingen av produkter bli effektivare för att snabbare komma ut på marknaden och till kunder. Bland annat kommer automatiserade tester att bli nödvändiga för att kunna ta emot täta interna leveranser, detta arbete är påbörjat. Men testmiljön måste bli mer lättillgänglig och flexibel för snabbare leveranser.

Syfte

Syftet med arbetet är att göra testning oberoende av systemspecifik hårdvara och därmed mer lättillgänglig genom att kombinera flera av Saabs existerande simulatorer för att bygga en testmiljö för mjukvara.

Mål

Målet med arbetet är att ta fram ett förslag på en simulerad testmiljö som kan testa mjukvara av delsystem för att avlasta de testriggar som används för systemtestning av GlobalEye Airborne Early Warning & Control system.

Frågeställningar

Studien kommer behandla följande frågor:

- Hur bygger man upp en simulerad testmiljö för test av mjukvara på systemnivå?
  - Hur har andra gjort?
  - Vad säger litteraturen?
  - Var börjar man?
  - Vad är viktigt att tänka på?

Vidare kommer arbetet behandla följande frågeställningar:

- Vad kan man använda av det som redan finns utvecklat hos Saab?
  - Hur kan man integrera dessa för att bygga en lättillgänglig testmiljö?
- Vad är för- och nackdelarna med de olika koncepten?
- Hur kan det implementeras i praktiken?

Avgränsning

Studien kommer att fokusera på hur man bygger upp testning av mjukvara i en simulerad miljö, där systemet som ska simuleras kommer att vara radarsystem och dess omvärld. Undersökningen av praktisk tillämpning kommer endast att fokusera på de simulatorer som finns hos Saab idag samt hur man kan kombinera dessa för att konstruera en testmiljö som kan avlasta de befintliga testriggarna som används. Det system som kommer att undersökas är sensorsystemet för GlobalEye.

2 SYSTEMET I FOKUS

Systemet som undersöks i det här arbetet är GlobalEye Airborne Early Warning & Control System (AEW&C). GlobalEye används för luftburen övervakning av mål på marken, luften och till sjöss. Den kan med hjälp av Erieye ER Radar upptäcka mål på avstånd upp till 450km och används bland annat för att övervaka gränser. Spaningssystemet är ett komplext system som sammankopplar flera delsystem. Det består av flera sensorer, två radarsystem, självförsvarssystem och kommunikationssystem, se figur 2.1.

Figur 2.1 - Global Eye systemöversikt

Systemet inkluderar följande sensorer :

- Erieye Extended Range Radar – Långdistansradar för luft, mark och sjömål
- Maritime Surveillance Radar – Sjöradar
- IFF – Igenkänningssystem av flygmål och andra farkoster (**I**dentification, **F**riend or **F**oe)
- AIS – Identifieringssensor för fartyg med AIS transponder (**A**utomatic **I**dentification **S**ystem)
- ADS-B – Positioneringssensor baserad på satellitnavigation (**A**utomatic **D**ependent **S**urveillance - **B**roadcast)
- ESM/ELINT – Signalspaningsenheter (**El**ectronic Signal **Int**elligence)
- EOS/IR Sensor – Elektrooptisk kamera (**E**lectro-**O**ptical **S**ensor)
- SATCOM/Datalink – Kommunikationssystem (**Sat**elite **Com**munications)

3 METOD OCH GEOMFÖRANDE

3.1 Metod

Arbetet har bestått av tre delmoment:

- En studie som är en teoretisk del för att få en grund inom ämnet och en bild av hur andra företag testar mjukvara i en simulerad miljö.
- En del som undersökte vilka typer av simulatorer som finns, hur de fungerar och hur testriggarna som används för systemtestning är konstruerade.
- En del som behandlade praktisk tillämpning och implementation av en testmiljö.

Studie

För att inhämta information till studien har litteratur som handlar om testning av mjukvara, testning av inbyggda system och testning av hårdvara används. För att ta reda på hur andra företag testar mjukvara i en simulerad miljö, så undersöktes först vilka kommersiella verktyg och metoder som kan användas för att bygga upp en testmiljö bestående av simuleringsmjukvara för att ersätta verklig hårdvara. Sedan fortsatte arbetet med att ta reda på vilka som använder dessa verktyg och vilka produkter företagen utvecklar för att i ett fortsatt arbete kunna undersöka hur dessa verktyg och metoder kan tillämpas tillsammans med de existerande simulatorer som finns hos Saab.

Tillämpning av testmiljö

Innan man kan fortsätta undersöka tillämpning av de simulatorer som finns hos Saab idag så ska studien vara klar för att enklare kunna avgöra vilken metod och vilka verktyg som skulle kunna användas för att bygga upp en testmiljö. För att ta reda på vad för typ av simulatorer som finns, hur de är uppbyggda och hur testmiljöerna man använder idag fungerar så hölls intervjuer med sakkunniga på Saab. Genom att teoretiskt undersökt hur man kan kombinera de olika simulatorerna med hjälp av de verktyg och metoder som framkommit under studien, har för- och nackdelar evaluerats.

Implementation

När studien och analys av teoretisk tillämpning var klar undersöktes huruvida en praktisk tillämpning av dessa var möjlig med de simulatorer, metoder och verktyg som har undersökts.

3.2 Genomförande

Förstudie

Vid val av testmetodik så undersöktes tre koncept Model In-The-Loop (MIL), Software In-The-Loop (SIL) och Hardware In-The-Loop (HWIL). Dessa valdes på grund av att de används vid testning av system där modellering och simulering av systemmiljö är i fokus.

Vidare så undersöktes fyra olika företag: National Instruments, MathWorks, dSPACE och Wind River. Företagen valdes på grund av att de är producenter och leverantörer av testriggar och mjukvaruverktyg som används i de tre tidigare nämnda testkoncepten. Genom att undersöka vilka som är kunder till de undersökta företagen så framgick det hur deras produkter tillämpas vid testning av mjukvara.

Parallellt med att systemet som var i fokus undersöktes, så kunde jag snabbt koppla samman hur de undersökta koncepten går att placera i verksamhetens utvecklingsprocess. Det resulterade i att MIL och HWIL inte längre var av samma intresse som tidigare då det är befintliga system som är i fokus samt att arbetet utreder hur de befintliga testriggarna kan avlastas genom att göra sig oberoende av systemspecifik hårdvara på systemnivå. Vilket varken MIL eller HWIL ger upphov till.

Intervjuer på Saab

Intervjupersonerna valdes i samtal med handledare på Saab. Personerna som intervjuades arbetar nära testmiljön som undersöks och har god insikt i hur systemen, simulatorer och registersystem fungerar. Intervjuerna hölls informella utan inspelningar. Några av frågorna som ställdes var:

- Hur testar man systemet idag?
- Vad finns det för för-och nackdelar med detta?
- Vad finns det för flaskhalsar i processen?
    - Hur påverkar dessa leveranser av systemet?
- Vad anser du vara viktigt att kolla på för att se hur man kan göra testningen mer lättillgänglig?

Intervjuerna gav kunskap om befintliga simulatorer och kännedom om systemet som testas framkom. Det framkom även hur man testar systemen och vad systemen består av. Då vissa av frågorna som ställdes var subjektiva så varierade resultatet på dessa. Det beror på att personerna som intervjuades arbetar inom flera områden. Men något som var av vikt, som tydligt framgick och verifierade bakgrunden till arbetet var att testriggarna som används idag är en trånga resurser. Det visade sig även att systemet som är i fokus består av delar som inte går att testa mjukt. Exempel på en sådan del är radarantennen, den måste testas skarpt.

Implementation av testmiljö

Efter intervjuerna så avgränsades fokus på systemet och fokus riktades mot implementation av en testmiljö som ska testa signal- och databehandlingsmjukvaran (SDF) och radaroperatörernas användargränssnitt **C**ommand and **C**ontrol som förkortas C2. Denna avgränsning gjordes på grund av de simulatorer och den hårdvara som fanns att tillgå. I samtal med handledare på Saab så gjordes bedömningen att en praktisk implementation var möjlig baserad på de tidigare nämnda faktorerna och den tidsbegränsning arbetet hade.

4 ANALYS

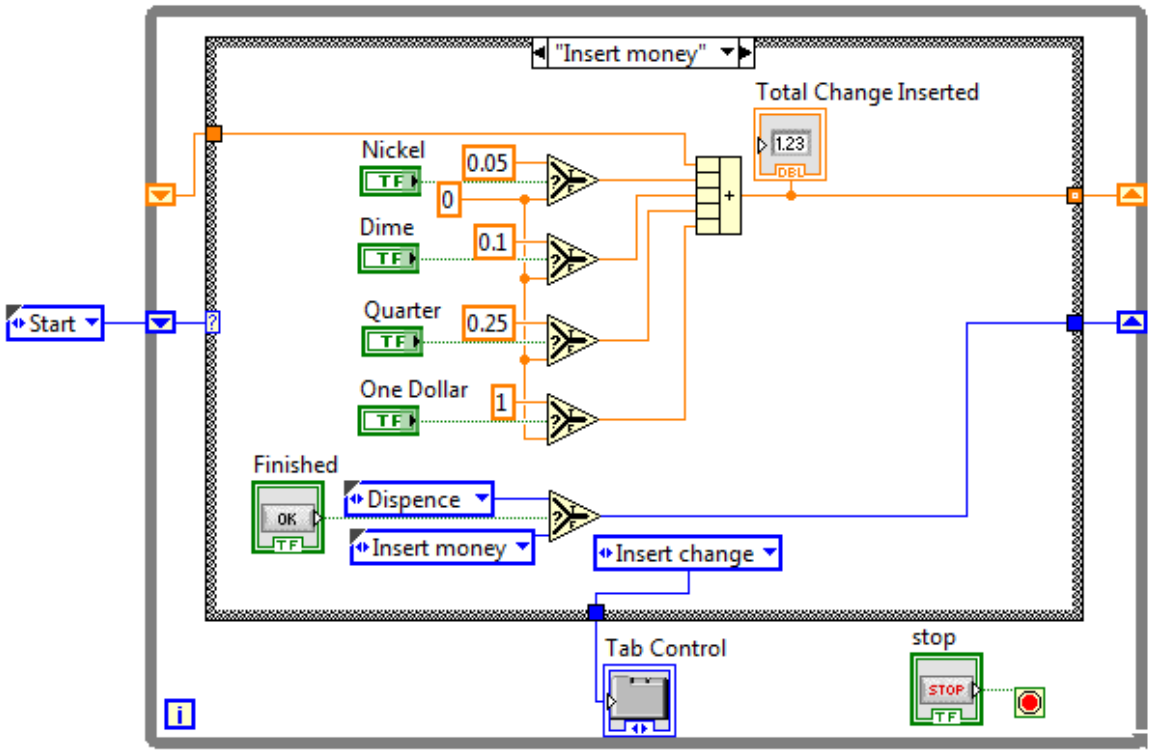
4.1 Förstudie

4.1.1 Testning i simulerade miljöer

För att kunna testa mjukvara utan tillgång till den fysiska hårdvaran så behöver man konstruera en modell av hårdvaran i en modelleringsmiljö. Det finns tre koncept som kommer att redovisas dessa är Model In-The-Loop (MIL), Software In-The-Loop (SIL) och Hardware In-The-Loop (HWIL).

MIL

Model In-The-Loop (MIL) innebär att en system-modell och dess miljö körs simulerat i ett modelleringsramverk, utan någon systemspecifik hårdvara. Modelleringen görs ofta med hjälp av grafisk programmering vilket innebär att man designar systemet på högnivå. Inom bilindustrin används funktionella modeller som inte tar robusthet och prestation i beräkning det vill säga att man endast testar modeller och ingen utvecklad mjukvara som körs på den verkliga hårdvaran. Modellerna kan användas tillsammans med kodgeneratorer för att automatiskt generera kod från system-modeller för att skapa en systemstruktur och en kodbas.



Figur 4.1 – Grafisk programmering i LabView [1]

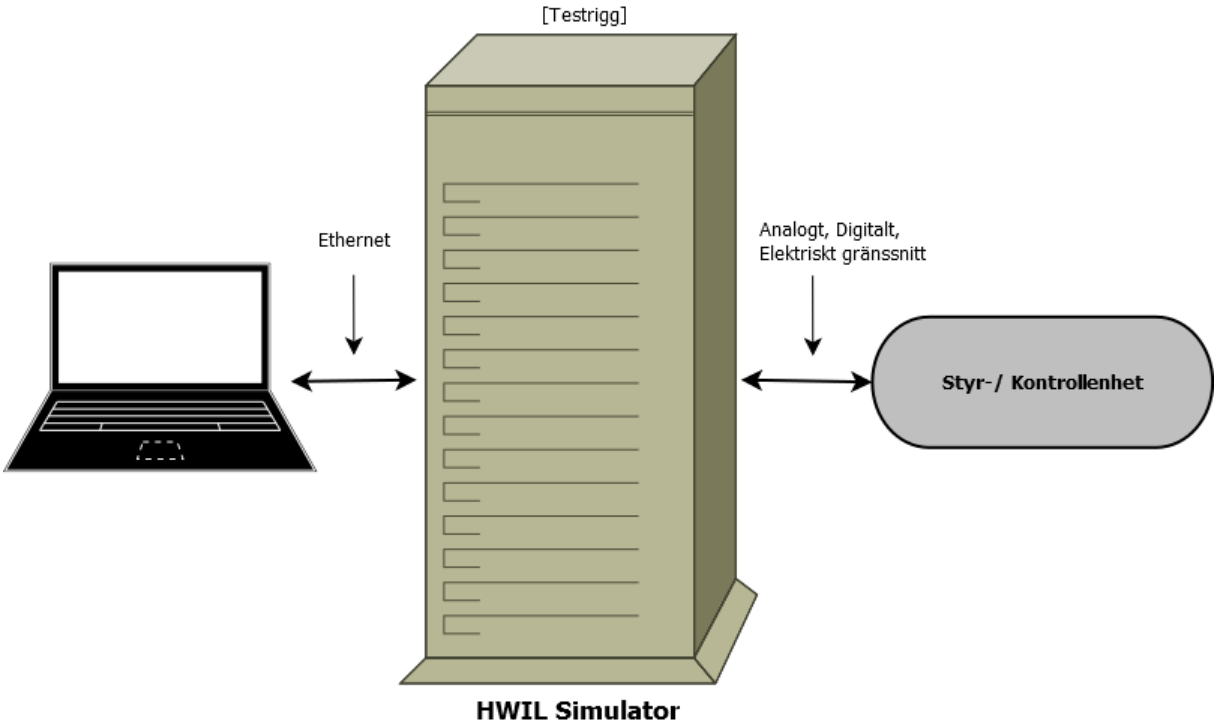
SIL

Software In-The-Loop (SIL), är en metodik då man kör mjukvara i en simulerad miljö oberoende av hårdvara. SIL kan ses som ett nästa steg från MIL ur ett utvecklingsperspektiv. Det som skiljer dessa åt är att man istället för att testa system-modeller med hjälp av grafisk programmering testar mjukvara i form av kod, som kan vara genererad utifrån modellerna eller egenutvecklad. Testningen sker utan användning av någon systemspecifik hårdvara som till exempel sensorer, mekaniska eller hydrauliska komponenter.

HWIL

Hardware In-The-Loop (HWIL), är en teknik för att testa komplexa inbyggda system. En HWIL testrigg består av:

- Utvecklingsdator (vanlig PC)
- Hårdvara som kör simuleringsmiljön (HWIL simulator)
- Hårdvara som ska testas (ECU/inbyggt system/styrenhet/kontrollenhet)



Figur 4.2 - HWIL systemets tre primära komponenter [2]

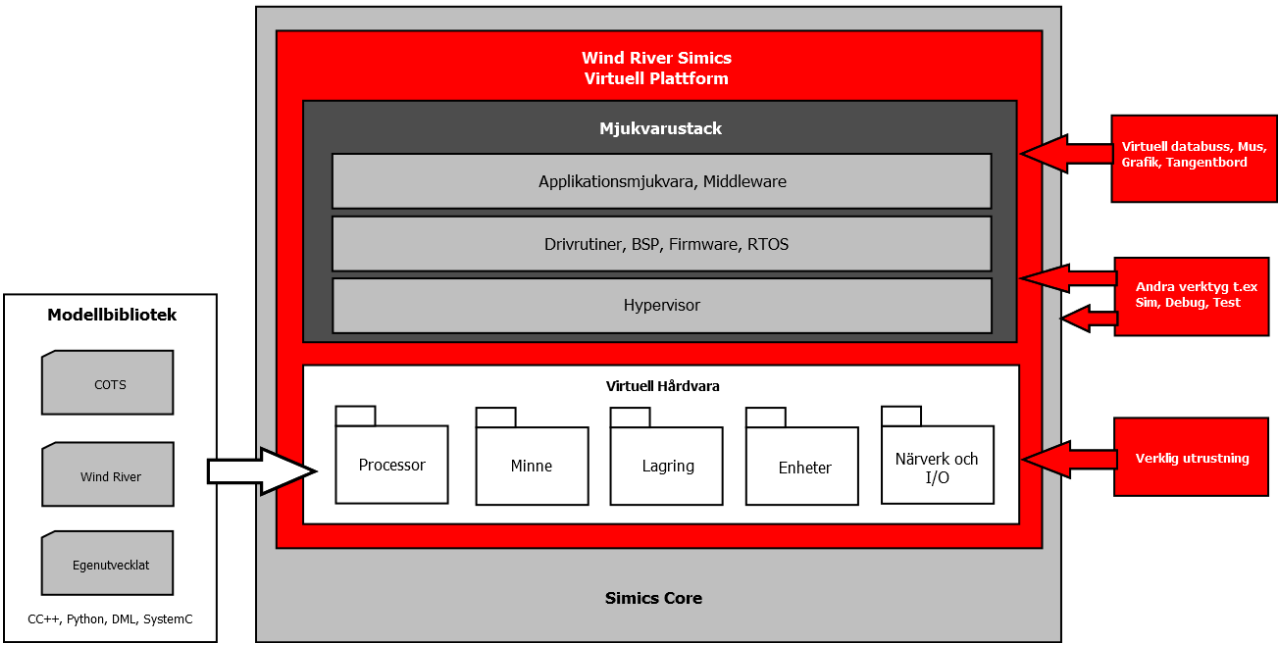
Testsystemet fungerar så att man kopplar in en enhet som kör den riktiga mjukvaran som ska testas till testriggen som kör simuleringsmiljön. Testningen kontrolleras genom utvecklingsdatorn. I HWIL simulatorn så konstruerar man de modeller som enheten interagerar med. Modellerna består av matematiska representationer av den miljö som enheten interagerar med i det verkliga systemet.

I dag finns det flera företag som utvecklar testriggar och mjukvara där man tillämpar testmetodikerna. Exempel på några är:

- Wind River
- National Instruments
- dSPACE
- MathWorks

Wind River

Wind Rivers simulator heter Simics och utvecklades från början av det svenska bolaget Virtutech som sedan blev uppköpt av Intel och Wind River. Mjukvaran används för simulering och modellering av hårdvara samt implementation och testning av mjukvara. Verktyget har ett brett bibliotek med många färdiga modeller som representerar komponenter som används inom fordon- och rymdforskningsindustrin. Bland annat MIL-STD-1553 databuss, CAN, FlexRay och flertalet andra viktiga komponenter som behövs vid uppbyggnad av komplexa system. Simics tillåter användaren att köra system i binärkod och har funktionalitet för att pausa exekvering och spara ner tillstånd för vidare analys. Den har även funktionalitet för att exekvera en instruktion i taget på maskinnivå.



Figur 4.3 : Simics Arkitektur [3]

Simics har stöd för flera CPU-arkitekturer och flertalet andra komponenter som krävs vid uppbyggnad av en dator. Användaren får möjlighet att skräddarsy den plattform man vill att mjukvaran ska köras på. Plattformen körs sen virtuellt på en dator. Det möjliggör att man kan köra flera plattformar parallellt på en dator och på så vis minskar behov av hårdvara.

National Instruments

National Instruments (NI) är ett amerikanskt bolag som är grundat i Austin, Texas och har sitt Sverige kontor i Kista, Stockholm. De utvecklar hård- och mjukvara riktat mot mjukvaruutveckling, testning och analys av inbyggda system. Ett marknadsområde som NI riktar sig mot är flygindustrin, både civil- och försvarsmarknaden.

NI:s HWIL-testrigg består av flera komponenter. Den konstrueras med deras egenutvecklade hårdvara, Commercial Of The Shelf (COTS) hårdvara och de kör även deras egenutvecklade mjukvara LabView. LabView används för grafisk programmering. Den har funktionalitet som gör att man kan testa, mäta, styra och analysera data. I LabView programmerar man på högnivå utan fokus på konstruktion av kretskort [4].

Systemen har stöd för fel-injektion, test av multi-ECU (Electronic Control Unit) system samt flertal gränssnitt för att koppla in olika styr- och kontroll enheter. De kallar sin simulator för *Turnkey simulator* vilket innebär att de bygger ihop all hårdvara som täcker de behov och krav som en kund har. De integrerar även mjukvaran åt kund vilket gör att man som användare ska kunna använda simulatorn utan att behöva leta efter hårdvara eller mjukvara för att få riggen att fungera.

MathWorks

MathWorks är ett amerikanskt företag som grundades 1984 i Kalifornien. Företaget utvecklar mjukvara som är inriktat mot avancerade matematiska beräkningar. Två av deras största produkter är Matlab och Simulink. Matlab är en utvecklingsmiljö med ett eget programmeringsspråk. Den används för att bland annat utföra avancerade matematiska beräkningar, konstruktion av användargränssnitt och konstruktion av grafer för analys av data. Simulink är en grafisk simuleringsmiljö för modellbaserad design som görs genom grafisk programmering. Båda programmen är vanligt förekommande i MIL- och HWIL testriggar som modelleringsverktyg.

dSPACE

Digital Signal Processing and Control Engineering (dSPACE) är ett tyskt företag som utvecklar verktyg för att testa, utveckla och kalibrera ECU:s för bil-,flyg-och automationsindustrin. De utvecklar både hård- och mjukvara. Ett av deras marknadsområden är HWIL testsystem.

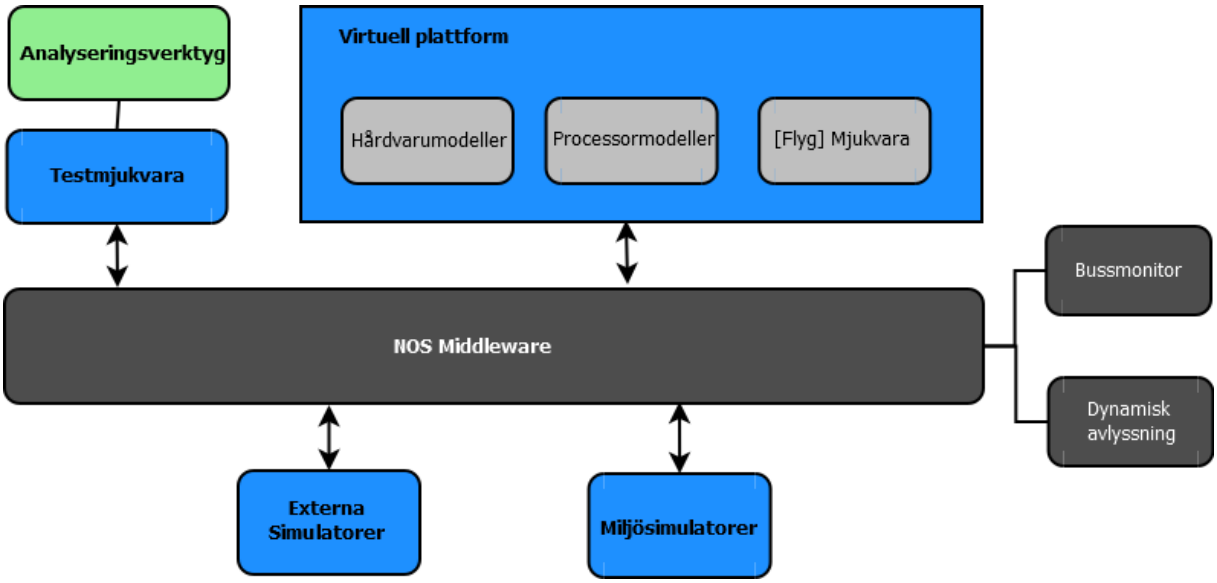
Testriggarna består av flera moduler och gränssnitt för inkoppling av ECU:er. De skräddarsyr testriggarna efter användarens behov. För testning och modellering så använder man VEOS [5] som är dSPACE egna mjukvara. VEOS- används för simulering, integration och testning av mjukvara i virtuella ECU:er. Modellering och konstruktion av testmodellerna görs med Simulink och Matlab.

4.1.2 Tillämpningar av simulerade miljöer

Simics

Ett företag som använder Simics är National Administration and Space Association (NASA) på deras avdelning för verifikation och validering (IV&V). De använder Simics tillsammans med egenutvecklade middlewares (MV) för att snabbare kunna konstruera system när hårdvara inte finns tillgänglig. Det gör att de snabbare kommer igång med utveckling och integration av system. En av fördelarna med Simics enligt Wind River och NASA är att de modeller man har konstruerat är återanvändbara vilket gör det till ett effektivt verktyg när nya system bestående av tidigare använda komponenter ska konstrueras [6]. Genom att använda Simics som simuleringsmiljö så behövs inte en ny simulator för varje nytt projekt som skapas, utan bara nya modeller av systemet [7]. NASA använder sig inte bara utav Simics utan de använder sig även av simulatorer som de själva har utvecklat. Exempel på en sådana är NASA Operational Simulator (NOS) och GO-SIM som är influerade av Simics.

NOS är en simuleringsarkitektur som används i deras simuleringsmiljöer. NOS har en MV som används för att porta flera simulatorer in till NOS för att simulera ett komplett system. NOS har likt Simics funktionalitet som gör att användaren kan pausa en exekvering och spara ner tillståendet för att senare kunna återstarta och analysera det. Den har även funktionalitet som gör att man kan injicera-,modifiera- och stoppa data, för att skapa fel som är svåra att skapa med den verkliga hårdvaran. Figuren nedan illustrerar hur NOS används, se figur 4.4.



Figur 4.4 - NOS implementation [8]

Arkitekturen används i syfte för att utveckla en software-only simulator med återanvändbar miljö för att enkelt skapa nya testmiljöer. NOS hjälper NASA att snabbare bygga upp testmiljöer som inkluderar kommersiella och statliga produkter som kör skräddarsydd programvara. NASA anser NOS vara en unik och konkurrenslös lösning [9].

National Instruments

Ett företag som använder NI:s produkter är Saab AB. Saab använder produkterna i syfte för att hitta en COTS lösning för att ersätta deras gränssnitt till deras skräddarsydda system till line-replaceable units (LRU:s) i deras simulatorer för JAS gripen. Det löste man genom att använda sig av NI:s Switch Load and Signal Conditioning (SLSC) system [10]. Syftet med att använda dessa produkter var att bryta testkurvan och reducera testkostnaden med 20%. Det gör man genom att ersätta in-house lösningar med kommersiell hårdvara på grund av bland annat modulariteten [11].

MathWorks

Volvo Cars är en användare som har använt MathWorks produkter för att utveckla en Human In-The-Loop simulator som körs i realtid. Volvo har konstruktioner som använder sig av avancerade hydrauliska system som har flera delkomponenter och subsystem som måste testas. Detta ansågs vara en kostsam process att utföra på fysiska prototyper. Därför valde man att modellera och simulera detta med hjälp av Simulink, Simscape och Simulink Real-Time på ett Virtual Memory System (VMS). Syftet var att skapa en miljö där alla subsystem kunde köras och att vara synkroniserade till samma realtidsklocka. De implementerade mjukvaran på en HWIL-rigg som var konstruerad med hårdvara från SpeedGoat. SpeedGoat är ett företag som är grundat av före detta anställda på Mathworks och specialiserar sig på realtidssystem.

Simulink användes i syfte för att skapa en modell av en motor som sen integrerades med subsystem för att representera ett komplett system. Genom att kombinera Simulink och Simulink Control Design så skapade man en kontrollmodell och körde closed-loop simulationer. Matlab användes för att presentera resultatet av simuleringarna. Efter att ha skapat och kalibrerat modellerna med dess parametrar genererade man C-kod med hjälp av Simulink Coder. Som sen kördes samverkande på tre olika COTS datorer från SpeedGoat.[12]

4.1.3 Leverans av mjukvara

För att kunna leverera mjukvara snabbare så är en viktig aspekt hur man testar sin mjukvara. I det här avsnittet redovisas för- och nackdelar med automatiserad testning och leverans av mjukvara.

Leverans av mjukvara brukar vara ett spänt tillfälle. Det beror på den risk som är associerad med processen. För vissa är processen manuellt intensiv. Det beror på att miljön där mjukvaran körs ofta är byggd separat. När miljön byggs upp är det flera installationer och konfigurationer som måste stämma till exempel servrar, referensdata och så vidare. Det gör att de är viktigt att varje steg implementeras rätt. Annars kan det sluta med att mjukvaran inte går att köra.

Anti-mönster som sällan är bra:

- Tillit på manuell testning för att förlita sig på att applikationen fungerar som den ska
- Leverans av mjukvara som tar mer än några minuter
- Frekventa korrigeringar av leveransen precis innan släpp

Leveranser går mot att bli helt automatiska. Det ska finnas två steg för att leverera mjukvaran vidare till utveckling eller produktionstest. Det är att välja version och miljö och sen klicka på "deploy". Att leverera mjukvara ska bara bestå av en automatiserad process som skapar installationen. Men alla är inte övertygade, här är några exempel på varför automatisering kan anses som nödvändig:

- När leverans inte är automatiserade sker alltid något fel. Frågan är bara om det är signifikant eller inte
- Om leveransen inte är automatiserad är den inte repeterbar eller pålitlig, vilket kan bli tidsödslande
- En manuell process måste dokumenteras varje gång. Men en automatiserad har sin egen dokumentation i form av sin kod. Vilket gör att automatiseringen alltid kommer vara up-to-date
- En manuell process kan inte garantera att instruktionerna har följts korrekt. Men det gör en automatiserad vilket utesluter fler mänskliga fel

Anti-mönster - Leverans till produktionslik miljö, när utveckling är klar:

- Antingen är produktionsmiljön för dyr att använda eller så är den för strikt kontrollerad eller så finns den inte klar i tid eller så har man inte gjort någon
- Utvecklingsteamet sammanställer alla installationsfiler, migrerar databaser, och leveransfiler till de som faktiskt ska genomföra leveransen - allt otestat i en miljö som liknar produktion eller "staging" [13, s.41]

Hur uppnår man målen?

Syftet är att leverera användbar, fungerande mjukvara så snabbt som möjligt. Tidsaspekten är viktig eftersom att utveckling kostar pengar. Så det man vill göra är att reducera cykel-tiderna. En viktig del av användbarhet är kvalité. Men kvalité betyder inte perfektion. Men ett mål ska alltid vara att leverera kvalité som ger värde för dess användare. Kort cykeltid , hög kvalité och mer frekventa och automatiserade leveranser.

Automatiserad:

- Om man varje gång bygger, levererar, testar manuellt så är processen inte repeterbar. Det gör att processerna runt omkring såsom dokumentation måste göras om. Det finns

även stora mänskliga faktorer som kan påverka processen vilket gör att man inte kan garantera att det görs rätt varje gång. Vilket betyder att man inte har kontroll över det

Frekvent:

- Om leveranserna är frekventa blir tiden mellan leveranserna kortare vilket minskar riskerna med processen samt snabbare feedback

Fungerande mjukvaruapplikation kan delas upp i fyra komponenter:

1. Exekverbar kod
2. Konfiguration
3. Värdmiljö
4. Data

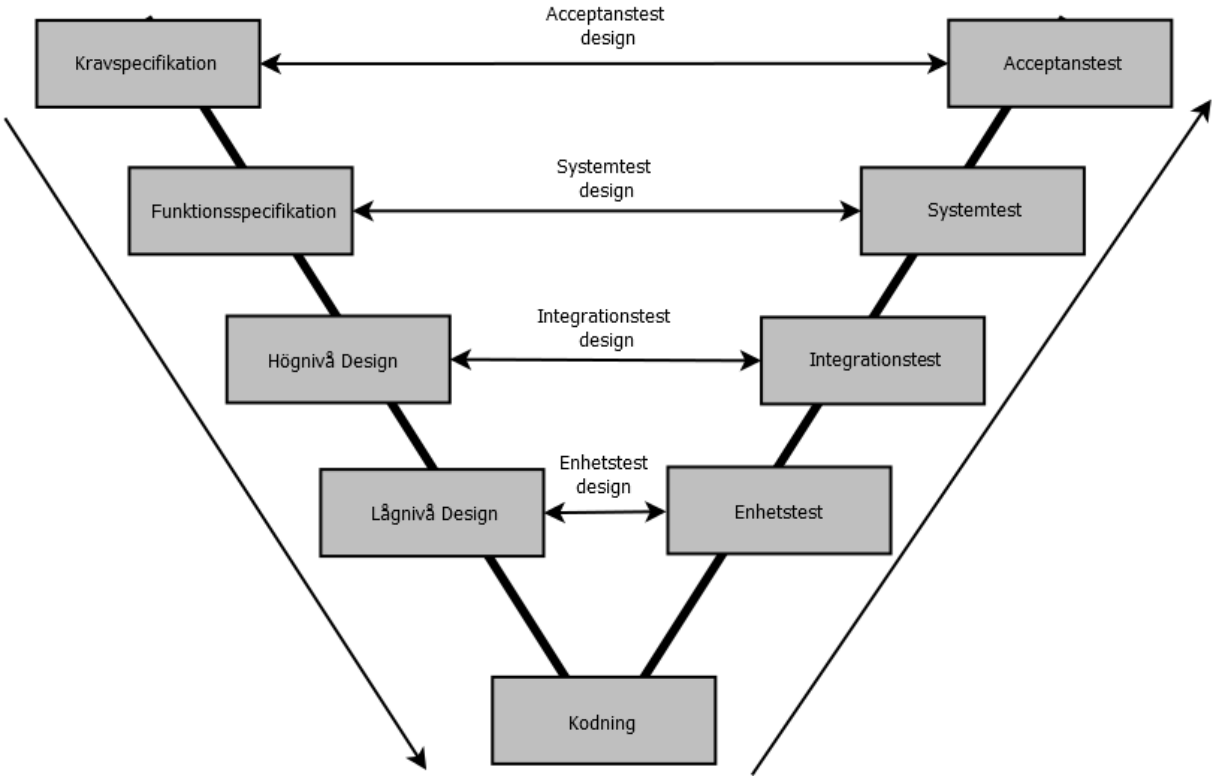
Om någon av komponenterna ändras kan funktionaliteten av applikationen ändras. Så varje gång någon av komponenterna ändras måste komponenten verifieras. Så om koden ändras bör det binära testas automatiskt [13, kap. 3]. Det som skiljer miljöerna åt ska belysas och räknas in till konfiguration [13, kap. 2]. Om miljön dit applikationen levereras ändras, ska hela systemet testas i den nya miljön [13, kap. 11]. Ändras strukturen av data, måste även detta testas [13 kap. 12].

Att automatisera tester är inte alltid rätt val att göra, följande punkter bör tas i åtanke [14]:

1. Ska testet köras flera gånger?
2. Kräver körning av testet support från fler än en person?
3. Hur mycket tid kan man spara?
4. Är testet på en kritisk punkt?
5. Är kraven som funktionen verifierar låg risk, stabil eller osannolikt påverkas av en förändring?
6. Är funktionen en gemensam faktor som blir en del av ett gemensamt resultat?
7. Är funktionen mottaglig för mänsklig påverkan?
8. Är funktionen tidskrävande? Innehåller den signifikanta vänteperioder mellan inaktivitet mellan teststegen?
9. Är funktionen trivial/högt repeterbar?
10. Kräver funktionen specialkompetens och är kompetensen att köra funktionen troligt att efterfrågas av andra?

4.1.4 Utveckling av mjukvara

Vid utveckling av mjukvara används traditionellt vattenfallsmodellen för att illustrera olika de utvecklingsfaserna. En modell som används för att betona hur de grundläggande nivåerna av testning speglar de tidigare faserna är V-modellen, se figur 5.2.



Figur 5.2: V-modell utvecklingsflöde [15]

Förklaring av de olika utveckingsfaserna:

- Kravspecifikation – Specificering av krav man ställer på systemet, vad systemet ska göra.
- Funktionsspecifikation – Specificering av systemets funtionalitet, vilka funktionaliteter som ska finnas.
- Högnivå design – Design av systemet på högnivå, vilka delar systemet ska bestå av.
- Lågnivå design – Design av hur systemets kodarkitektur ska se ut.
- Kodning – Utveckling av kod baserat på lågnivå designen.
- Enhetstest – Testning av en enskild del av systemet på lågnivå.
- Integrationstest – Testning av integration mellan en del av systemet på högnivå.
- Systemtest – Testning av systemet mot funktionsspecifikationen.
- Acceptanstest – Testning av systemet mot kravspecifikationen.

4.2 Fortsatt arbete på Saab

4.2.1 Intervjuer på Saab

Efter att förstudien var klar så hölls intervjuer med personer som är arbetar med de testsystem som används idag. Intervjuerna hölls informella utan inspelningar. Några av frågorna som ställdes var:

- Hur testar man systemet idag?
- Vad finns det för för-och nackdelar med detta?
- Vad finns det för flaskhalsar i processen?
- Hur påverkar dessa leveranser av systemet?
- Vad anser du vara viktigt att kolla på för att se hur man kan göra testningen mer lättillgänglig?

Personer som intervjuades:
Person A – Utvecklare av simulator, SYSIM (Scenariogenerator)
Person B – Utvecklare av delsystem, SDF (Signal- och databehandling)
Person C – Erfaren testare och systemingenjör, SYSIM (Scenariogenerator)
Person D – Systemingenjör, SYSIM/C2 (Scenariogenerator/radaroperatörernas användargränssnitt)
Person E - Utvecklare plattform och erfaren testare, automatiska tester
Person F – Specialist registreringssystem

Under intervjuerna framkom information kring vilka system som används vid testning och hur testprocessen ser ut.

4.2.2 Verktyg som används idag

När man testar hela systemet så används flera simulatorer och verktyg. Bland annat för att simulera sensorer, radarmål och för att registrera och analysera data. I det här avsnittet redovisas några simulatorer och verktyg som har en central uppgift i testsammanhang för radar på Saab.

SYSIM

SYSIM är en scenariogenerator där man kan skapa en omvärldsmiljö. Med hjälp av SYSIM så kan man skapa olika radarmål så som sjö- och flygmål. Användaren kan specificera vad målen ha för egenskaper fart,höjd, riktning, storlek med mera. SYSIM används när man inte testar radar skarpt.

RES

Radar Environment Simulator (RES) är en del av SYSIM. RES uppgift är att skapa frekvensdata som beskriver det scenario som en användare har skapat i SYSIM. Den används

för att ersätta Exiter Receiver (EXR) vilket är den enhet som kommunicerar med radarantennen i det riktigt systemet och bland annat skickar styrsignaler till antennen.

NERES

New Erieye Recording and Evaluation System (NERES) är en inspelningsenhet som registrerar datatrafik från de gränssnitt som skickar data mellan de olika delsystemen. Den används för evaluering och visualisering av data i realtid.

Reafil

Realfil är en internutvecklad applikation som används för att analysera och översätta den data som NERES har spelat in under testning. Reafil kan även användas i realtid för analys av data. Den används också för att kontrollera NERES.

C2

Command and Control (C2) är ett MMI som används av radaroperatörerna. Systemet används för att visa detekteringar från de olika sensorerna, styra de olika sensorerna, skapa sökområden och kommunicera mellan operatörer.

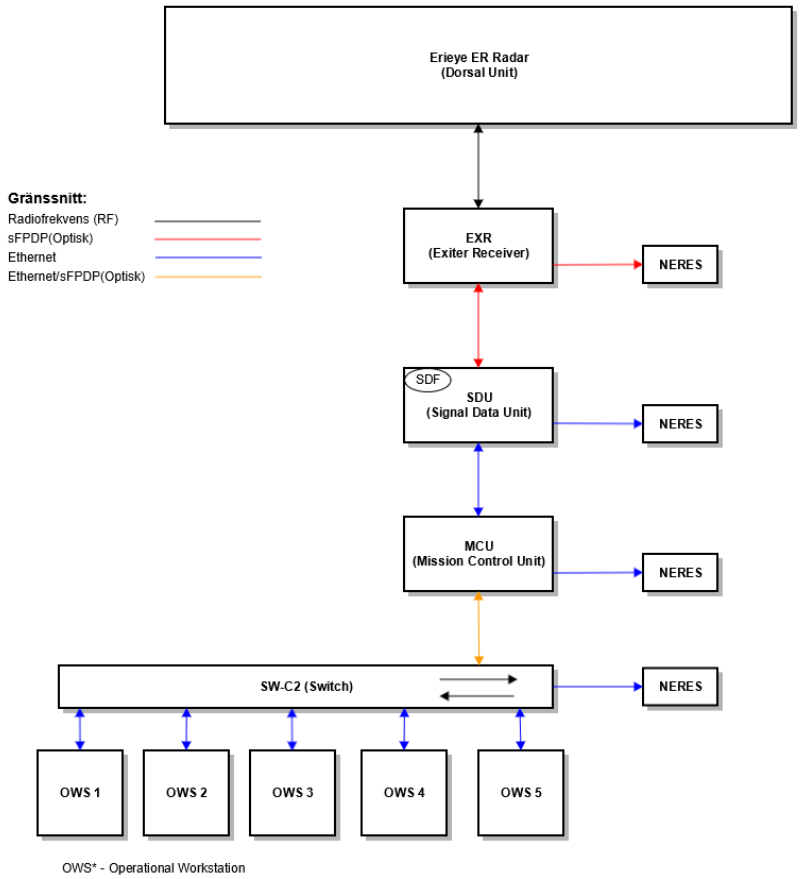
MTS

Mission Training System (MTS) är ett system som används för träning av radaroperatörer. Systemet har ett antal operatörsplatser och fyra stycken instruktörsplatser. Syftet med systemet är att träna radaroperatörer. Systemet tillåter instruktörer att skapa scenarion och planera uppdrag. I den här miljön har C2 en central roll då det är användargränssnittet som operatörerna interagerar med. Systemet kan ses som en testmiljö av C2 då funktionaliteten av användargränssnittet används men omvärlden simuleras.

4.2.3 System under test

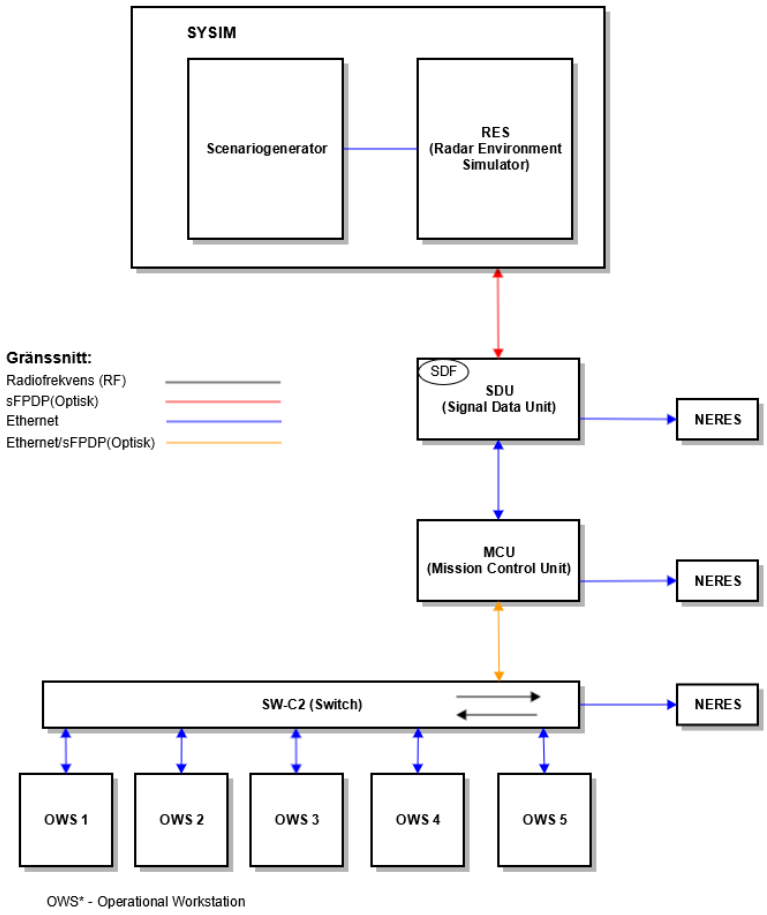
Systemtestning av radar sker skarpt med det riktiga systemet där omgivningen kan simuleras med hjälp av SYSIM. Innan systemen kommer till testriggen så har delsystemen testats individuellt på respektive instans. När systemet testas så skickar varje delsystem sin data till NERES som registrerar datatrafik och kapslar in det för att spara ner till fil. Genom att sedan analysera det registrerade datat med hjälp av Reafil så kan man se hur varje delsystem har svarat. Nedan illustreras två testriggar som används, leveransriggen och referensriggen, se figur 4.5 och 4.6.

Figur 4.5 visar en grov översikt på hur leveransriggen är konstruerad. Den består av Erieye ER radar vilket är den riktiga radarantennen, Exiter Receiver (EXR) enheten som styr radarn, Signal Data Unit (SDU) där signal- och databehandlingen sker, det vill säga där radarmål identifieras genom att information som ges av sensorerna analyseras. Mission Control Unit (MCU), SW-C2 är en switch som kommunicerar med Operational Workstation (OWS) där radaroperatörerna sitter och interagerar med systemet genom C2.



Figur 4.5: Leveransrigg grov översikt

Leveransriggen består av den riktiga mjukvaran körandes på den riktiga hårdvaran. Det är här man får en komplett bild av hur systemet fungerar innan flygprov där man testar systemet i dess ideala miljö.



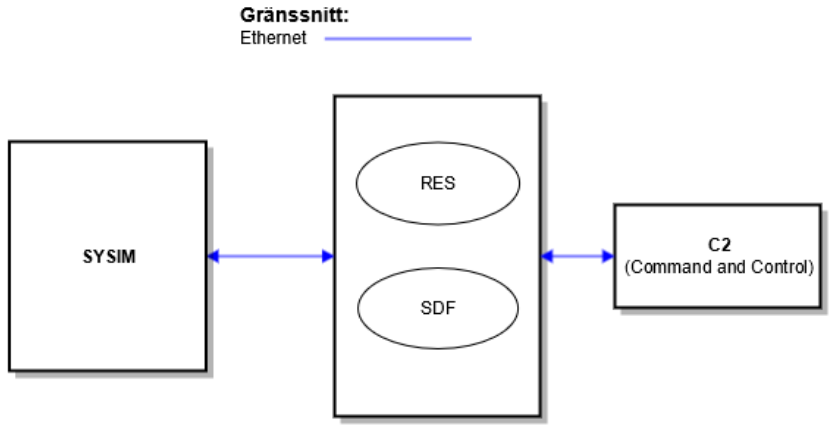
Figur 4.6: Referensrigg grov översikt

Referensriggen består av delvis riktig hårdvara och riktig mjukvara. Vilket innebär att vissa delar är simulerade. Till skillnad från leveransriggen så simulerar man sin omgivning, Erieye ER radar och EXR med hjälp av SYSIM och RES.

5 FÖRSLAG PÅ TESTMILJÖ

Under intervjuer med sakkunniga framgick det att man idag testar signal- och databehandlingen (SDF) mjukt. SDF används för att identifiera mål utifrån det data som samlas in med hjälp av de sensorer och radar som finns i systemet. Dessa mål visualiseras sedan i radaroperatorernas användargränssnitt (C2). Men i testriggen för SDF så använder man sig av ett egenutvecklat gränssnitt vilket innebär att första integrationen med C2 sker i de befintliga testriggarna.

Ett sätt för att testa SDF mot C2 i en mindre testmiljö är att simulera omgivningen. För att kunna göra detta så behöver man använda sig av SYSIM och Radar Environment Simulator (RES). Ett förslag på en sådan tillämpning illustreras i figur 5.1.



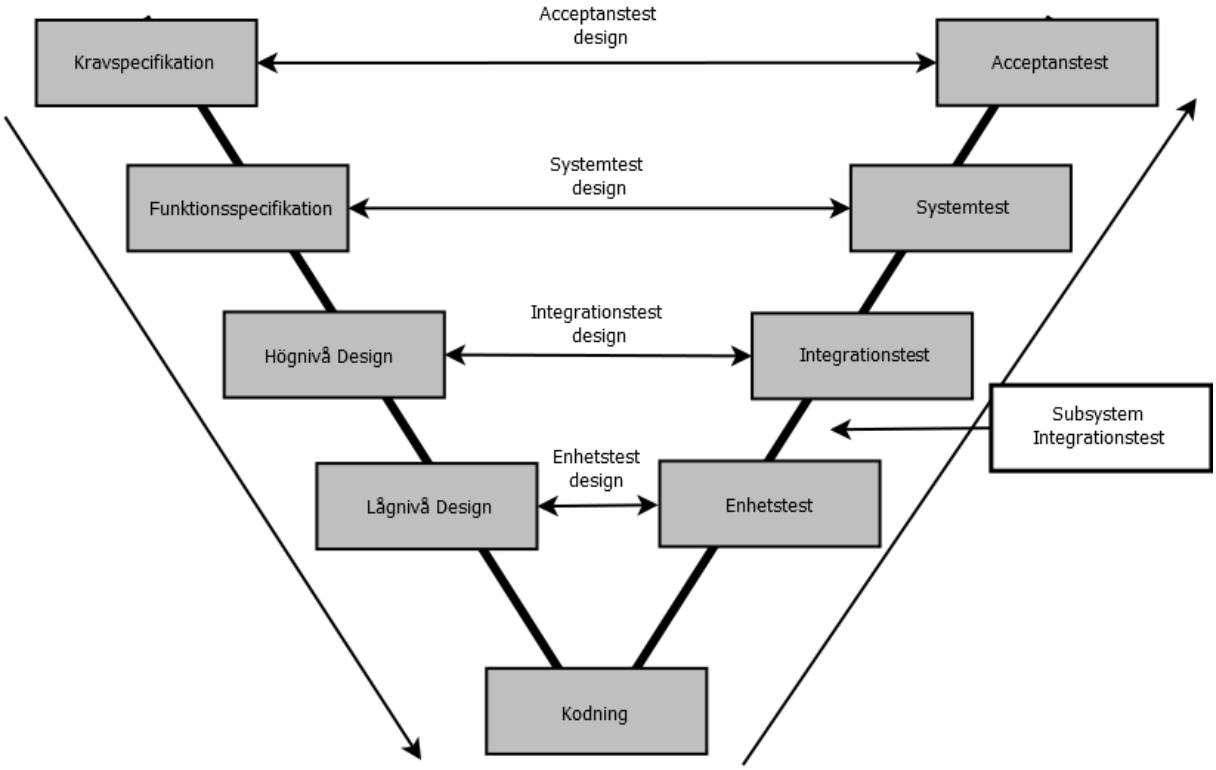
Figur 5.1: Förslag på testmiljö – delsystem grovskiss

Tillämpningen innebär att man sköter kommunikation mellan de olika delsystemen med hjälp av Ethernet. Det resulterar i att man gör sig oberoende av systemspecifika gränssnitt så som seriellt optiskt data protokoll (sFPDP). Man använder sig endast av en operatörsplats vilket innebär mindre åtgång av hårdvara. Tillämpningen innebär också att man skulle köra RES och SDF mjukvaran på kommersiell hårdvara.

5.1 Utvecklingsprocess

För att förstå hur den förslagna testmiljön skulle kunna användas i ett utvecklingsarbete så måste man förstå i vilken ordning olika delar av ett system testas. Kollar man på hur utvecklingsarbetet ser ut idag så kan man se att utvecklingsflödet liknar V-modellen.

Om man integrerar SDF och C2 i ett tidigare skede blir modellen enligt figur 5.2. Där testning av SDF och C2 blir ett subsystem integrationstest. Det innebär att man kan upptäcka eventuella fel innan man integrerar SDF mot C2 i en komplett systemrigg. Testriggarna för systemtestning är trånga resurser. Vilket resulterar i att det här teststeget gör att testriggarna avlastas då testningen inte behöver utföras i ett komplett system.

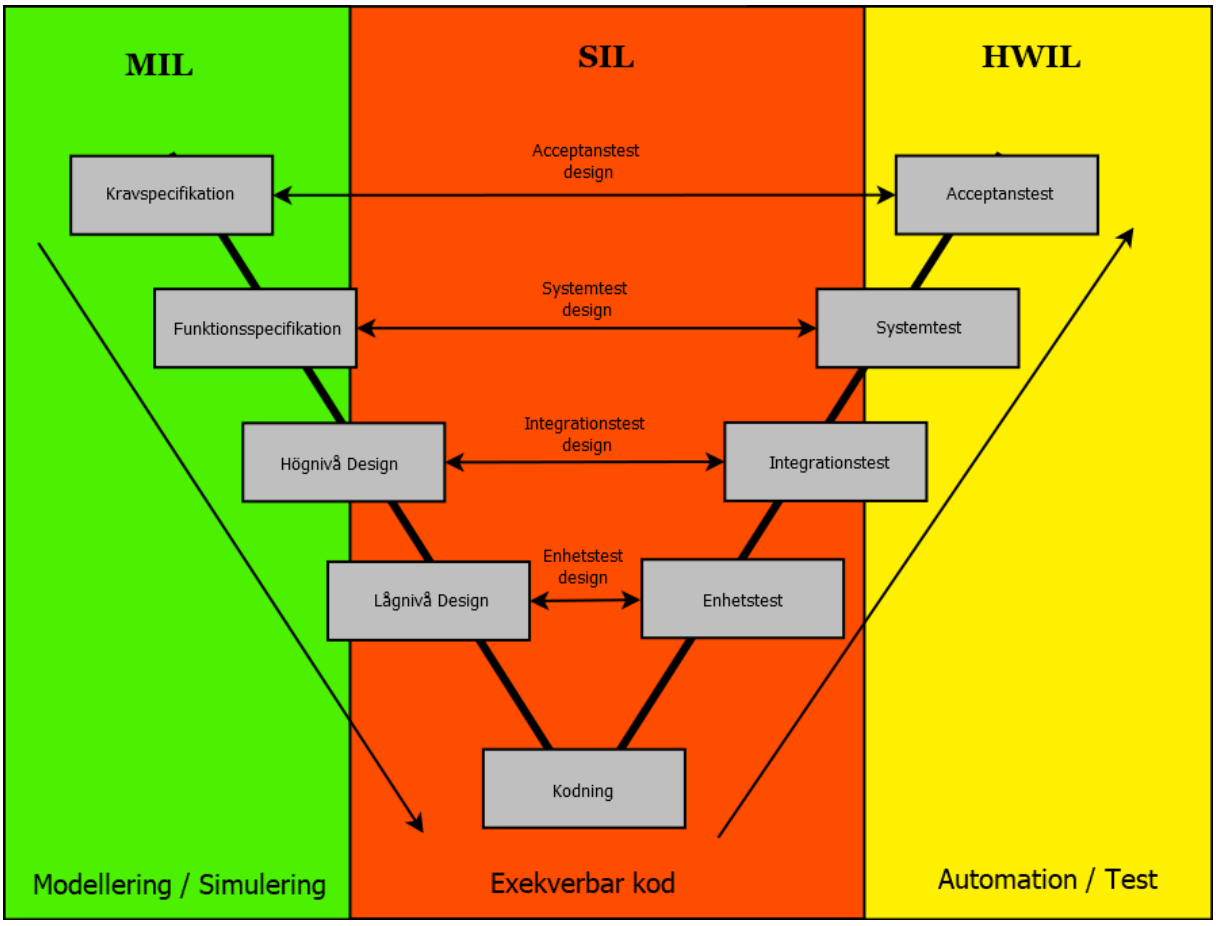


Figur 5.2: V-modell kompletterad med subsystem integration

Ser man till hur V-modellen är konstruerad så kan man utläsa att: Ju längre i testprocessen man kommer, ju dyrare blir ofta felen att åtgärda. Genom att testa ytterligare ett steg innan full integration så blir det billigare och enklare att åtgärda eventuella fel som kan uppkomma. Viktigt att tänka på är:

- Går sub-integrationen att isolera så att testningen får ett värde?
- Är testmiljön kostnadseffektiv? (är miljön dyrare än vad kostnaden av att åtgärda eventuella fel i nuläget är)

Om man skulle placera in de redovisade koncepten Model In-The-Loop (MIL), Software In-The-Loop (SIL) och Hardware In-The-Loop (HWIL) i en V-modell, så skulle man få en modell som ser ut enligt, se figur 5.3.



Figur 5.3: V-modell med MIL, SIL och HWIL

MIL skulle kunna placeras i början av arbetsflödet där man bygger modeller för att kunna testa konceptet på hög nivå. Genom att sen generera kod så fortsätter man med SIL. När mjukvaran har testats med hjälp av SIL. Så kan man använda sig av HWIL och köra mjukvaran på riktig hårdvara mot en testrigg som simulerar systemets omgivning. Sista steget i processen innebär att man kör acceptanstest i en komplett systemrigg.

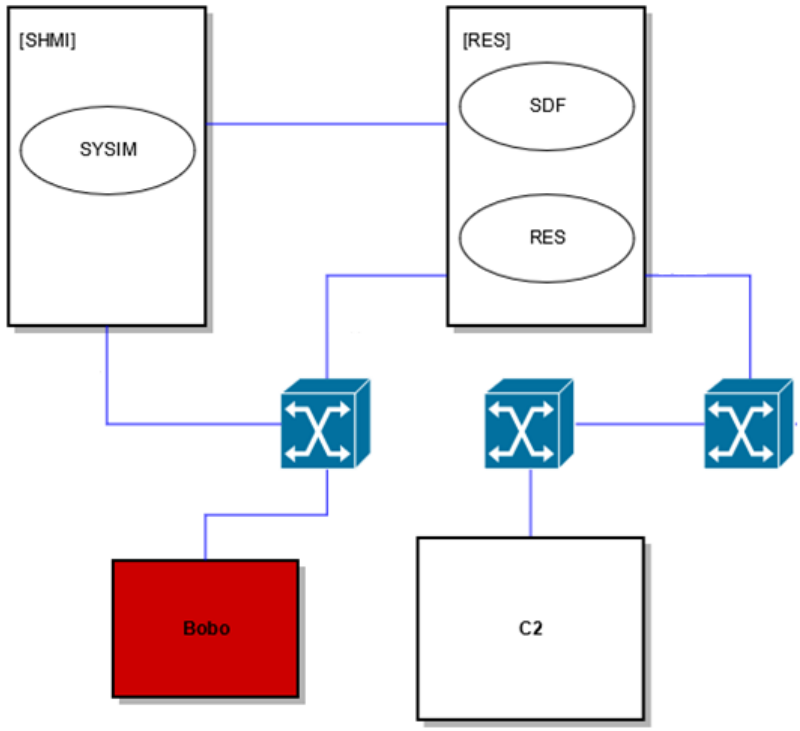
I förslaget där man integrerar SDF mot C2 så är inte MIL eller HWIL av intresse. Det beror på att integrationstestning sker i en annan del av utvecklingsprocessen. I den presenterade miljön så använder man sig av simulatorer och annan mjukvara för att realisera miljön.

6 IMPLEMENTATION AV TESTMILJÖ

Som nämnt i tillämpningsavsnittet så finns det ett delsystem som man kan testa mjukt som resulterar i att integration och testning av signal- och databehandlingen (SDF) och radaroperatörernas användargränssnitt (C2) blir smidigare och som bidrar till att systemriggarna blir avlastade. Genom att ha undersökt hur man skulle kunna göra en sådan implementation så undersöktes vilken hårdvara som fanns att tillgå och vad för teknikaliteter som behövde lösas för att realisera tillämpningen. Implementationen som redovisas i det här kapitlet är ett ”proof of concept”. En avgränsning som gjordes var att fokus inte ligger på prestanda. Det beror på att den hårdvara som användes var hårdvara som fanns att tillgå för att göra ett provskott av testmiljön på Saab.

Implementationen som genomfördes var att koppla samman SYSIM, radar environment simulator (RES), SDF och C2, enligt den förslagna tillämpningen. Syftet med implementationen var att skapa en testmiljö som kan avlasta systemriggarna och vara lättillgänglig samt att testa SDF mot C2 i ett tidigare skede.

För att isolera den här delen av systemet och konstruera en testmiljö bestående av befintlig hårdvara och mjukvara så behövde samplingstakten som SDF-mjukvaran bearbetar data i sänkas. Genom att sänka samplingstakten så blev SDF mjukvaran mindre resurskrävande vilket resulterade i att SDF och RES kunde köras på samma dator. Testmiljön implementerades enligt figur 6.1.



Figur 6.1: Arkitektur på implementerad testmiljö

Systemet fungerar enligt följande; C2- och SDF mjukvaran laddas in genom Bobo som är en gateway in i nätverket där miljön körs. Gateway används för att C2- och SDF mjukvaran utvecklas i andra miljöer. Sedan skapas ett scenario med hjälp av SYSIM. Scenariot körs igång och ethernet kommunikation till RES upprättas. RES omvandlar scenariodatat som skapas med hjälp av SYSIM till frekvensdata som ska representera riktiga radarsignaturer för målen. SDF analyserar frekvensdatat och skickar sedan information om de detekteringar som har upptäckts till C2 som visar informationen för operatören.

C2 mjukvaran som körs i testriggen är omodifierad och dagliga versioner finns att tillgå. Vilket innebär att man kan köra C2:s nyaste version. Eftersom att utvecklarna av SDF själva laddar in sin mjukvara så är även den av den nyaste versionen. Bobo kan även användas till att fjärrstyra testmiljön vilket gör den lättillgänglig och resulterar i att fysisk närvaro för att använda testmiljön inte behövs.

6.1 För- och nackdelar

Fördelar:

- Man kan i ett tidigare skede testa integrering mellan SDF och C2
- Systemet kör de absolut senaste versionerna av SDF- och C2 mjukvaran vilket de befintliga testriggarna inte kör
- Man kan testa nya funktioner mellan C2 och SDF snabbare, vilket gör att man får snabbare återkopplingsloopar
- Miljön är lättillgänglig och kräver inga systemspecifika gränssnitt
    - Testriggen består delvis av kommersiell hårdvara
- Miljön går att köra fjärrstyrd

Nackdelar:

- För att använda miljön så krävs det att man ändrar sitt sätt att arbeta
    - SDF mjukvaran behöver packas om för att kunna köras i testriggen
- Miljön använder sig inte av något ramverk som kör automatiska tester
- Hårdvaran som kör SDF behöver mer prestanda
- SDF- och C2 mjukvaran laddas in manuellt

7 RESULTAT

Måluppfyllnad

Målet med arbetet var att presentera ett förslag på en simulerad testmiljö som kan avlasta de befintliga testriggarna för systemtestning. Jag anser att målet uppfylldes då ett förslag på en testmiljö har presenterats och dess koncept har testats. Genom att använda testmiljön så får man ett till steg i testprocessen vilket resulterar i att mjukvaran för signal-och databehandling och C2 kan testas i ett tidigare steg innan systemtestning. Det leder i sin tur till att en mer stabil mjukvara levereras till systemriggarna. En del fel som upptäcks vid systemtestning kan då eventuellt upptäckas tidigare i den förslagna testmiljön.

Förstudie

Hur bygger man upp en simulerad testmiljö för test av mjukvara på systemnivå?

Beroende på var i utvecklingsprocessen man befinner sig så lämpar sig olika testmetodiker. Om man är i början av en utvecklingsprocess så kan man använda sig av MIL-verktyg till exempel Simulink eller Matlab. Om man har färdig kod som kan exekveras så kan man använda sig av SIL verktyg. Ska man testa mjukvara på systemspecifik hårdvara så kan man använda sig av HWIL verktyg.

Hur har andra gjort?

Ett vanligt förekommande tillvägagångssätt är att man använder sig av kommersiella produkter, både när det gäller hårdvara och mjukvara. Beroende på var i utvecklingsprocessen man är så varierar valet av dessa. Ett annat alternativ är att man kombinerar kommersiella produkter med egenutvecklad hård- och mjukvara.

Vad säger litteraturen?

Det finns inte mycket information att tillgå gällande MIL,SIL och HWIL. Beroende hur utvecklingsprocessen ser ut så kan MIL, SIL och HWIL tillämpas i olika faser. Men när det kommer till testning generellt så är automatisk testning och leverans av mjukvara viktig.

Vad är viktigt att tänka på?

- Viktigt att tänka på när man bygger en testmiljö är att man vet vad det är man vill testa och vad för typ av testning det skulle innebära.
- Vad för mjuk- och hårdvara man har att tillgå. Om det är värt att bygga en alternativ testmiljö.
- Man bör också undersöka om den tänkta miljön har stöd för automatiserad testning och leverans av mjukvara.

Var börjar man?

- Man bör kartlägga vad man har att tillgå gällande mjuk- och hårdvara.
- Man bör kartlägga vilka verktyg man kan använda sig av för att simulera den testmiljö man vill att systemet ska köras mot.
- Man bör också ta reda på om dessa verktyg uppfyller de kraven man ställer på sin tänkta testmiljö, till exempel stöd för olika plattformar och modularitet.

Fortsättningsstudie
Vad kan man använda av det som redan finns utvecklat?

Av det som fanns utvecklat kan man använda sig av flera simulatorer och delsystem. Några av dessa är SYSIM, RES,C2 och SDF.

Tillämpning
Hur kan man integrera dessa för att bygga en lättillgänglig testmiljö?

Genom att kombinera SYSIM, RES, C2 och SDF så kan man konstruera en testmiljö som kan testa integrationen mellan C2 och SDF. Miljön kan också användas till att testa nya funktioner i SDF.

Implementation
Hur kan det implementeras i praktiken?

Genom att låta SYSIM och RES ersätta radarantennen och EXR så kan man låta dessa simulera omvärlden och styrning av radarantennen. För att sedan låta SDF bearbeta informationen och kommunicera med C2 för att testa dess funktioner.

Vad är för- och nackdelarna med implementationen?

Fördelar:

Det finns nu ett alternativ för att skapa snabbare återkopplingsloopar vid utvecklingsarbete och förslag på en testrigg som man har testat i praktiken. Testmiljön ger möjlighet till ytterligare ett teststeg för SDF och C2 innan systemtestning.

Nackdelar:

För att kunna använda testmiljön optimalt så behöver fler funktioner implementeras och hårdvaran där SDF körs måste uppdateras för bättre prestanda.

8 DISKUSSION

Förstudie

De resultat som har framkommit under förstudien baserar sig främst på information som fanns att tillgå på olika företags hemsidor där de distribuerar dokumentation som har med deras hård- och mjukvara att göra. Vetenskaplig litteratur som beskriver utvecklingsprocesser och omvärldsstudien blev stöttande material för att kunna placera testkoncepten i en utvecklingsprocess. De tre koncept som undersöktes var MIL, SIL och HWIL. Under arbetes gång så blev MIL och HWIL mindre intressant för det tillämpningsområde som hamnade i fokus inte berörde dessa koncept och då inte heller var intressanta att tillämpa.

Kollar man på hur andra företag har byggt upp sina testmiljöer så tyder de exempel som redovisas i arbetet på att man går mot användning av kommersiell mjuk- och hårdvara. Men beroende på tillämpningsområde så anpassar man miljön efter de behov som finns. Fokus på testriggarna har varit att det ska vara flexibla miljöer med stöd för flera plattformar och gränssnitt. Man har även fokuserat på att hålla nere antalet systemspecifika hårdvarukomponenter för att bygga billig lösningar.

Tillämpning av testmiljö

Vid undersökning huruvida en tillämpning av de undersökta koncepten var möjlig framkom det att en tillämpning var möjlig med de simulatorer som fanns att tillgå hos Saab. Den information som handlar om de delsystem och simulatorer som redovisas i arbetet har framkommit genom intervjuer med sakkunniga som arbetar nära miljön. Det innebär att stor del av den information som redovisas inte kan hittas genom att googla eller läsa någon publik litteratur. Tillämpningen som hamnade i fokus var sub-integration av signal- och databehandlingen (SDF) och radaroperatörernas användargränssnitt (C2). Med tanke på att dessa redan kördes på hårdvara som kunde användas så krävdes inga mer resurser. Under intervjuerna så framkom det att man tidigare har haft en tanke på att genomföra en sådan implementation men att det inte har planerats eller genomförts. Med tanke på komplexiteten av systemen så fanns det inte möjlighet att sätta sig in i hur systemen testas i detalj under arbetes gång. Det innebar att en realisering av testmiljön skulle innebära stor hjälp från personer som arbetar med systemen.

Implementation av testmiljö

Testmiljön implementerades på befintlig hårdvara som fanns att tillgå hos Saab. De simulatorer och mjukvara som användes var färdig mjukvara och inget som har utvecklats under arbetets gång. Det centrala i att få systemet att fungera var kommunikationen mellan de olika delarna. Eftersom att systemen inte har körts tillsammans på det här sättet tidigare så innebar det att nätverksadresser behövdes konfigureras och SDF mjukvaran anpassas. Det gjordes av sakkunniga som arbetar på Saab. Man kan se det som att examensarbetet var den drivande kraften för realisera implementationen.

Avslutningsvis för sammanfatta tankar om arbetet så kan man säga att: Det har varit svårt att veta var man ska börja när det kommer till hitta information kring testmetodiker och hur de har tillämpats av andra företag. Mycket av informationen som finns att tillgå redovisas ofta av företag där man kan få en känsla av att man vill marknadsföra sin produkt. När det kommer till det fortsatta arbetet på Saab så har all information som framkommit där varit information som inte går att hitta utanför företaget. På så vis har mina handledare spelat en betydande roll för att det är dem som har rekommenderat de personer som har intervjuats. För att kunna implementera den föreslagna tillämpningen så krävdes stor hjälp från sakkunniga, det beror på att det inte fanns tid att kunna fördjupa sig i systemen.

Om man ser på implementationen ur ett hållbarhetsperspektiv så innebär det att man gör sig mindre beroende av systemspecifik hårdvara. Det resulterar i att användningen av metaller som återfinns på kretskort inte behövs i samma utsträckning. En annan faktor som förändras är förbrukningen av elektricitet, genom att minska på antalet datorer som drivs av elekticitet så minskar förbrukningen.

9 SLUTSATS

Arbetet har resulterat i att en testmiljö som testar integration mellan SDF och C2. Miljön består av simulatorer och hårdvara som fanns att tillgå innan arbetets start. Testmiljön är ett proof of concept som visar att man kan testa SDF mot C2 i ett tidigare stadium än vad som görs idag. Genom att använda miljön så kan man skapa snabbare återkopplingsloopar vid utveckling av nya funktioner. Det bidrar till att man snabbare kan leverera mjukvara. Man kan även i ett tidigare stadium upptäcka eventuella integrationsfel vilket resulterar i felen blir billigare att åtgärda.

För att börja använda testmiljön i praktiken så bör man verifiera att testmiljön uppfyller de krav man har, det vill säga att den verkligen fungerar som den ska. Man bör också automatisera ompackning av SDF mjukvaran samt inladdning av all mjukvara för att spara tid.

Förslag till fortsatt arbete

- Använda bättre hårdvara så att SDF kan prestera idealt, det vill säga kunna köra med normal samplingstakt
- Köra hela testsystemet på en serverdator genom att köra all mjukvara på virtuella plattformar, likt Simics Simulator
- Implementera funktionalitet som laddar in den nyaste mjukvaran av C2 och modifierar SDF mjukvaran automatiskt så att den går att köra i testriggen utan någon handpåläggning
- Implementera ett ramverk för automatiserad testning som tillsammans med automatisk inladdning av SDF och C2 mjukvara kan köra automatiska tester vid varje ny uppdatering
- Se över om det finns andra delsystem som man kan inkludera i den nuvarande lösningen
- Se över om det finns andra delsystem där kan skapa liknande lösningar som kan innebära snabbare återkopplingsloopar

10 REFERENSER OCH LITTERATURFÖRTECKNING

[1] Wikipedia, LabVIEW State Machine example (Insert_Money_Case)
https://en.wikipedia.org/wiki/File:LabVIEW_State_Machine_example_(Insert_Money_Case).png
[Hämtad: 2018-01-21]

[2] National Instruments, Arkitekturbild. National Instruments, Corp,
Austin, Texas 2009,
http://www.ni.com/cms/images/devzone/tut/clip_image001_20091022140938.jpg
[Hämtad: 2018-10-02]

[3] Wind River Simics, produktblad Simics. Wind River Systems, Inc,
Alameda, Kalifornien 2015, https://www.windriver.com/products/simics/simics_pn_0520.pdf
[Hämtad: 2018-10-01]

[4] National Instruments, LabView. National Instruments, Corp,
Austin, Texas 2017, http://sweden.ni.com/fpga
[Hämtad: 2018-10-02]

[5] dSPACE, VEOS. dSPACE GmbH,
Paderborn 2018,
https://www.dspace.com/en/inc/home/products/sw/simulation_software/veos.cfm
[Hämtad: 2018-10-02]

[6] Wind River, NASA IV&V. Youtube, LLC,
San Bruno, Kalifornien 2013, https://www.youtube.com/watch?v=PDLSrXhu8gM
[Hämtad: 2018-09-25]

[7] Wind River Simics, NASA Meets Satellite Project Testing and Verification Goals with Wind River Simics. Wind River Systems, Inc, Kalifornien USA 2018,
https://www.windriver.com/customers/customer-success/documents/NASA_IVV_SS_0113.pdf
[Hämtad: 2018-25-09]

[8] NASA, Enabling Technology. National Aeronautics and Space Administration,
Washington D.C. 2017,
https://www.nasa.gov/centers/ivv/dynamic_analysis_enabling_tech.html
[Hämtad: 2018-10-01]

[9] NASA, Independent Test Capability. National Aeronautics and Space Administration,
Washington D.C. 2017, https://www.nasa.gov/centers/ivv/jstar/ITC.html
[Hämtad: 2018-10-01]

[10] Anders Tunströmer, Saab Elevates Testing of the World's Most Cost-Effective Fighter Plane. National Instruments, Corp, Austin, Texas 2018, http://sine.ni.com/cs/app/doc/p/id/cs-17491
[Hämtad: 2018-10-02]

[11] National Instruments, 04 NI Week 2016 Day 1 Gripen. Youtube, LLC,
Corp, Austin, Texas 2016, https://www.youtube.com/watch?v=hEks7TUyU-Q
[Hämtad: 2018-09-29]

[12] MathWorks, Volvo Construction Equipment Streamlines Product Development with a Real-Time, Human-in-the-Loop Simulator. MathWorks, Inc, Natick, Massachusetts 2016, https://www.mathworks.com/company/user_stories/volvo-construction-equipment-streamlines-product-development-with-a-real-time-human-in-the-loop-simulator.html?s_tid=srchtitle
[Hämtad: 2018-10-02]

[13] Humble Jez, Farley David. Continuous Delivery: reliable software releases through build, test, and deployment automation.Vol.1. Crawfordsville, Indiana; 2010

[14] Graham, Dorothy. Fewster, Mark. Experiences of test automation: Case studies of software test automation, Crawfordsville, Indiana; 2012 (s.569, figur 25.2)

[15] Jorgensen C. Paul. Software Testing: A Craftsman's Approach.Vol.4. Boca Raton, Florida; 2014 (s.208)