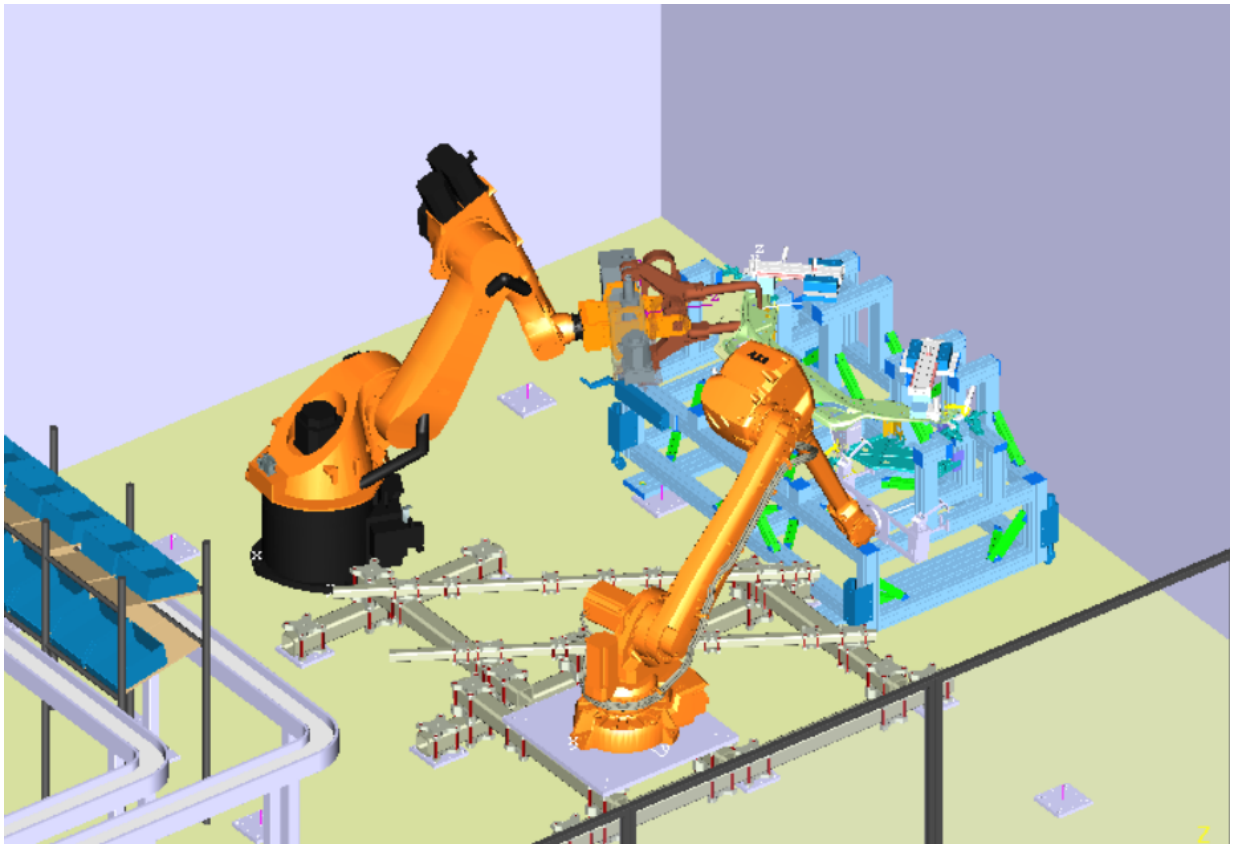
CHALMERS



Virtuell idrifttagning av tillverkningscell

Virtual commissioning

Kandidatarbete vid Institutionen för signaler och system

THANG HOANG
HENRIK JOHANSSON
ALEXANDER LYDH

Institutionen för signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Virtual Commissioning
A report about using virtual commissioning
in a manufacturing cell consisting of two industrial robots.
THANG HOANG, HENRIK JOHANSSON, ALEXANDER LYDH
Department of Signals and Systems
Chalmers University of Technology

Abstract

Virtual commissioning is when a manufacturing system is virtually created and used for testing the code used in controlling the system. There are several advantages in testing the code in a virtual production system before implementing it in the real process. Savings in investments and time are to be expected by commission the process virtually in early stages of the development of the manufacturing system. A lot of the tasks that are required for the development can be performed earlier and alongside the manufacturing and assembling of the manufacturing system.

In the project a virtual model of a manufacturing cell is created containing two industrial robots. Ten parts is assembled in a simulation to create the front section of a Volvo S80. The goal was to create a simulation of the production system and to virtually prepare the program code used. Virtual commissioning and virtual preparation as method was then evaluated. The project is limited to only validate the simulation and code against the created model, and not against the real manufacturing cell.

The method of simulating a manufacturing system is today widely spread in the industry. The simulations can be done both time and event based. Simulations involving virtual commissioning is done event based, which increases the functionality and makes it possible to transfer the program code from the simulation to the real manufacturing system.

To create the simulation two pieces of software is used, Process Designer and Process Simulate, both from Siemens.

Several conclusions about the usability of the evaluated working method can be drawn. Aspects that affect whether or not virtual commissioning is a suitable solution in developing a manufacturing system include, but are not limited to, the systems size and complexity.

For further evaluation of preparation of program code and virtual commissioning some changes in the projects scope are recommended. The next step should include different manufacturing systems of different sizes and complexity. The use of different commissioning and preparation methods should also be use to do a comparison of the methods.

Sammanfattning

Vid virtual commissioning, virtuell idrifttagning, simuleras ett produktionssystem virtuellt där programkod verifieras och valideras innan implementering i det verkliga systemet. Att testa sin programkod innan koden överförs till den fysiska utrustningen har flera fördelar. Det går att göra stora besparingar både tids- och kostnadsmässigt genom att i ett tidigt skede bereda produktionssystem virtuellt. Att arbeta virtuellt möjliggör att flera uppgifter kan utföras parallellt med arbetet med designen av ett produktionssystem, något som tidigare krävde ett färdigdesignat system.

En virtuell modell av en produktionscell med två industrirobotar har skapats och sedan har en ihopsättning av en Volvo S80-front, bestående av tio delar, simulerats. Målet var att skapa denna simulering och att virtuellt bereda tillhörande logik och programkod. Sedan har metoden med virtuell idrifttagning och virtuell beredning av programlogik utvärderats. Arbetet är avgränsat till att enbart validera simuleringen och programkoden mot det skapade modellen och inte mot den fysiska produktionscellen.

Metoden att simulera produktionssystem är idag utbrett. Simuleringar kan göras både tids- och händelsestyrda. Simuleringar som görs inom virtual commissioning är händelsestyrda. Det ökar funktionaliteten och gör det möjligt att överföra logik och programkod från simulering till det faktiska produktionssystemet.

För att skapa simuleringen har programvarorna Process Designer och Process Simulate från Siemens använts.

Flera slutsatser om den utredda arbetsmetodens användbarhet kan dras, exempelvis att det inte är lämpat för alla produktionssystem. Faktorer som påverkar huruvida virtual commissioning och att virtuellt bereda programkod är en lämplig metod att använda innefattar, men begränsas inte till, systemets storlek och komplexitet.

För fortsatt utvärdering av virtuell beredning av programlogik och virtual commissioning rekommenderas några ändringar av utvärderingens omfång. Nästa steg bör vara att undersöka olika produktionssystem med hjälp av flera olika beredningsmetoder, och sedan jämföra resultaten för att kunna dra bättre underbyggda slutsatser.

Förord

Denna rapport är ett resultat av kandidatprojektet *SSYX02-14-09 Virtual Commissioning* vid Chalmers tekniska högskola under våren 2014. Projektets omfattning är 15 högskolepoäng.

Vi i projektgruppen vill tacka:
Petter Falkman, vår handledare vid Chalmers som hjälpt och stöttat oss genom projektet.
Andreas Jonsson, vår kontaktperson på Volvo Cars som hjälpt oss mycket med programvaran.
Per Nyqvist, för hjälp med konvertering av CATIA-filer till JT-format.

Författarna, Göteborg 19 Maj 2014

Innehåll

Förkortningar

ABB	Robottillverkare
CAD	Computer Aided Design
CATIA	Computer Aided Three dimensional Interactive Application
CTH	Chalmers tekniska högskola
HMI	Human-Machine Interface
JT	Filformat för 3D-produkter
KUKA	Robottillverkare
OPC	Objekt Linking and Embedding for Process Control
PD	Process Designer
PLC	Programmable Logic Controller
PLM	Product Lifcycle Management
PPU	Institutionen för produkt- och produktionsutveckling på Chalmers
PS	Process Simulate - PLM-programvara
PSL	Production Systems Laboratory - Produktionssystemslaboratoriet på Chalmers
VC	Virtual Commissioning - Virtuell idrifttagning
VCC	Volvo Car Corporation - Volvo Personvagnar

1

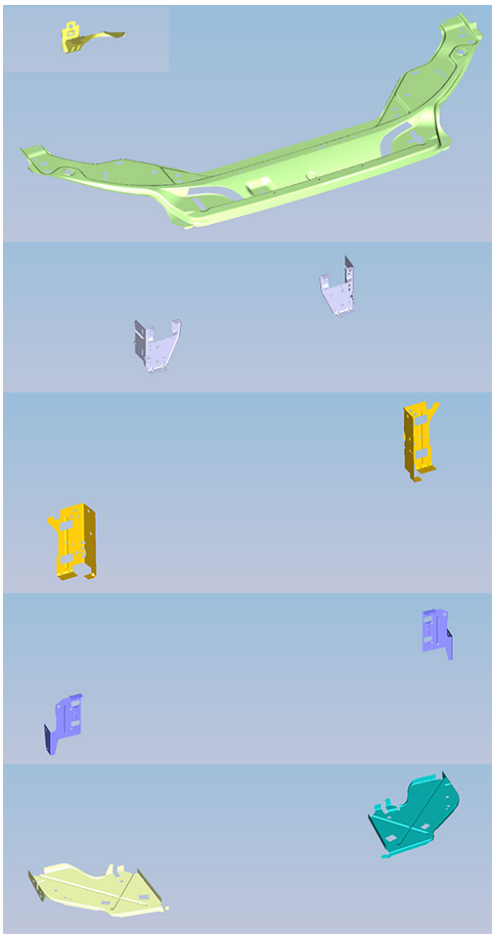
Inledning

Att simulera ett produktionssystem i datorn, helt virtuellt, och sedan testa färdig programkod kallas virtuell idrifttagning eller virtual commissioning (VC). Istället för att designa ett produktionssystem och sedan anpassa logiken till designen går det att med hjälp av VC att styra och förbättra designen med hänsyn till styrlogiken. En styrlogik som går att ta tillvara på och fortsätta att använda i det verkliga systemet istället för att ta fram en ny kompatibel kod. Vid användning av VC vid framtagningen av processer kan tester utföras och fel hittas redan innan produktionssystemet är på plats i fabrik.[1]

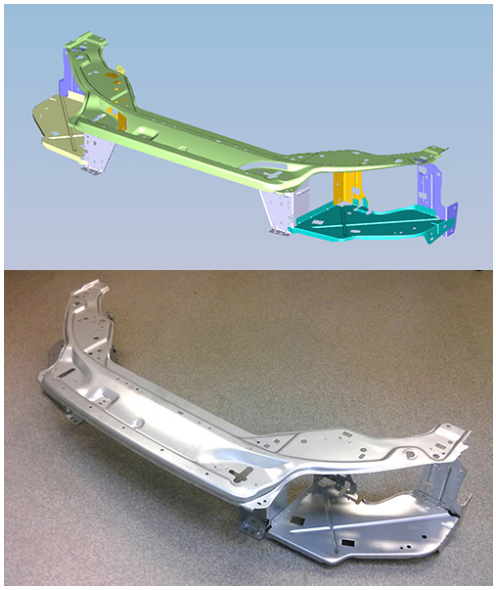
Att testa sin programkod innan koden överförs till den fysiska utrustningen har flera fördelar. Stora besparingar både tids- och kostnadsmässigt går att göra genom att i ett tidigt skede bereda produktionssystem virtuellt. Det är hård konkurrens i tillverkningsindustrin och en snabbare time-to-market ger marknadsfördelar. En högre kvalitet på produktionen är även möjlig att uppnå. Att använda VC har inte bara fördelar, utan kräver en del resurser, så som kompetens och licenser till programvaror.[1]

1.1 Problembeskrivning

En virtuell modell av en produktionscell med två industrirobotar skall skapas och sedan skall en ihopsättning av tio delar, figur 1.1, som tillsammans bildar en Volvo S80-front, figur 1.2, simuleras. Logik och programkod till simuleringen skall samtidigt tas fram. Frontdelarna kommer att lyftas in av industrirobotarna för att sedan punktsvetsas samman till en front. Arbetet är virtuellt och kommer inte att utföras i den fysiska världen. Programvaror som kommer att användas är CATIA för modellskapande, Process Designer för uppställning av det virtuella produktionssystemet och Process Simulate för programmering och simulering av produktionssystemet. Arbetsmetoden som används vid skapandet av styrkoden och den virtuella modellen ska sedan utvärderas.



Figur 1.1: Alla tio frontdelar till Volvo S80



Figur 1.2: Ihopsatt front till Volvo S80

1.2 Bakgrund

Det finns mycket resurser att spara genom att välja att bereda en process innan testkörning av den i en fysisk miljö. Även om arbetssättet skiljer sig mycket mellan olika företag är det generellt mycket dubbelarbete som görs, då två olika instanser tar fram styrvillkor och logik. Simuleringsfasen har tidigare utnyttjat mycket styrlogik för att framställa en trovärdig simulering, men mycket av detta arbete har sedan kasserats då logiken har varit svår att få tillgång till för utnyttjande i det fortsatta arbetet. Detta berättade Andreas Jonsson från VCC, 2014-03-21.

Vidare berättade Andreas att tidigare har simuleringar utförts enligt sekventiella körscheman, där simuleringen följde ett Gantt-schema. Detta var ett stort problem då denna programmering av en process skiljer sig markant från hur verkligheten ser ut. Ett styrprogram är oftast händelsestyrt och det är denna typ av styrning som skall utvärderas.

Vissa studier visar på att så mycket som 25% av tiden som en konstruktionsingenjör lägger på ett projekt finns i idrifttagningsfasen. Så mycket som 15% spenderas på att rätta fel som uppstått i programmeringen av styrlogiken. Dessa förluster syns i projekt som inte har implementerat virtuell idrifttagning. Det finns alltså stora resurser att spara om virtuell idrifttagning används.[1]

I och med en ständig snabb teknisk utveckling krävs det oftast att en tillverkningsprocess behöver läggas om, då nya produkter tas fram[2]. Andreas från VCC förklarade att det därför läggs stor vikt på företag idag att kunna hålla en viss flexibilitet för ändringar i sitt produktionssystem. En flexibilitet som bör utnyttjas är att i så stor grad som möjligt kunna planera större ändringar, exempelvis av ett produktionsgolv, så att ändringen kan genomföras med så kort produktionsstopp som möjligt.

I över tio år har utveckling av möjligheten att virtuellt bereda tillverkningssystem undersökts, och Volvo Cars har nyligen satt igång en satsning på detta område. En av de drivande krafterna bakom denna vilja att hitta ett system för att virtuellt bereda en tillverkningsprocess är de besparingar företag kan göra med metoden.[3] Samtidigt kan virtuell beredning användas för att enklare möta många av de krav som dagens marknad ställer på många produkter. Dessa marknadskrafter inkluderar, men begränsas inte till, en kortare livscykel, minskade materialkostnader och en snabbare "time-to-market". [4] För att minska time-to.market vid utveckling av nya maskiner använde Tetra Pak sig av virtuell simulering. Tidigare var utvecklingsprocessen på Tetra Pak sekventiell och i huvudsak grundad på fysiska tester. Med virtuell simulering kunde tiden som behövdes för att bygga och testa fysiska prototyper reduceras.[5]

Vid ett tidigare arbete vid produktionslaboratoriet (PSL) vid CTH har lyckad simulering och kommunikation med produktionscellen gjorts med programvaran DELMIA och en OPC-server. OPC-servern används för att DELMIA skall kunna kommunicera med cellens PLC och därmed överföra programmeringen och simuleringens beteende till den verkliga produktionscellen för verifiering. [6]

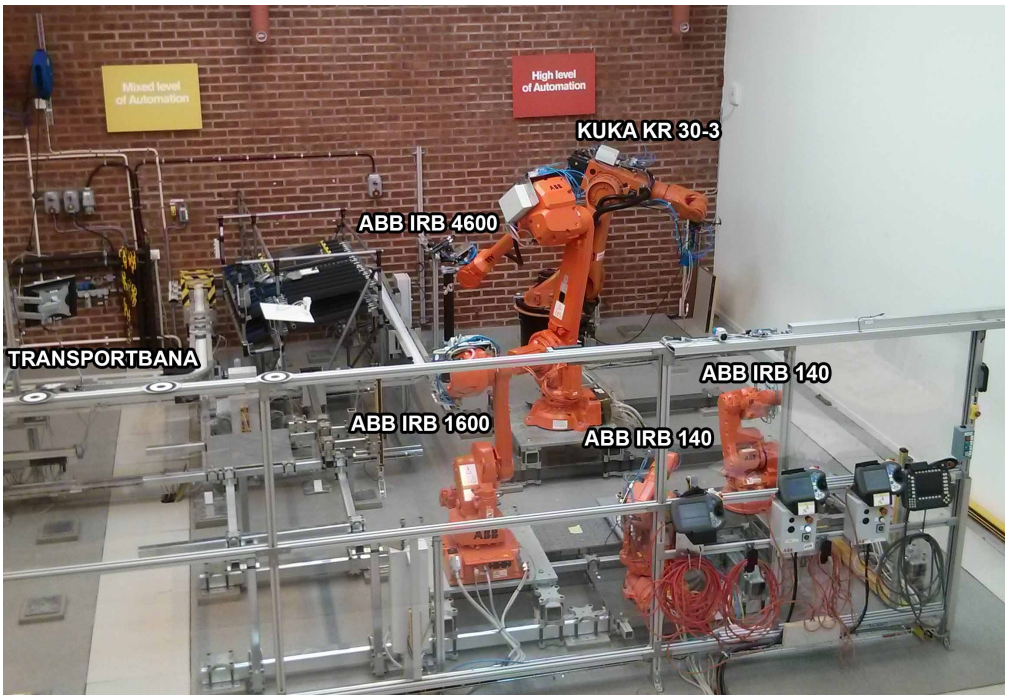
Vid forskning om ingenjörsmässiga metoder för tillverkning av adaptiva tillverkningssystem användes VC för att virtuellt utforska nya lösningar för mekatronisk optimering. Metoden applicerades sedan vid nyutveckling av en modul till en tillverkningslina som tillverkar sensorer till bilindustrin.[7]

Fokus i projektet kommer att ligga på programmeringen av produktionscellen i PSL. PLC- resp. robotprogram kommer att implementeras i programmet. Inga ändringar av robotarnas och övrig befintlig utrustnings placeringar kommer att göras. Uppställning av ny virtuell utrustning i den virtuella cellen kommer att göras där utrymme finns och så att den är funktionell. Verktyg som saknas kommer att konstrueras och modelleras upp, fast enbart en enklare modell av dem.

1.3 Produktionscellen

Den produktionscell som använts som grund för simuleringarna är beläget i PSL, se figur 1.3. I cellen finns det fem robotar, fyra ABB-robotar och en KUKA. ABB-robotarna är en IRB 4600, en IRB 1600 och två IRB 140. KUKA-roboten är av modell KR 30-3. Se appendix **A** för specifikationer.

Robotarna är inte utrustade med verktyg för att sätta ihop en front till en Volvo S80. Något svetsaggregat finns ej och det finns inga planer på att utrusta dem med det på grund av bl.a. brandsäkerhet och ventilationsproblem, berättade Petter Falkman vid en intervju den 9 maj 2014. Gripverktyg finns, men de är inte anpassade till produtionscellen och används därför inte. Någon fixtur till S80-delarna finns ej heller.
I cellen finns även en transportbana. Denna produkttransportör kommer att användas till att transportera in de nio mindre delarna av fronten. Anledningen till att den största frontdelen inte transporteras in med hjälp av transportbanan är för att delen är för stor.



Figur 1.3: Produktionscellen i PSL

1.4 Syfte

Projektets syfte är att kunna bereda programlogik virtuellt samt kunna genomföra en virtuell idifttagning i ett tidigt skede av beredningsfasen.

1.5 Mål

Projektets mål är att utvärdera användandet av virtuell beredningsteknik och en tidig virtuell beredning av logik, vid framtagningen av en tillverkningsprocess. Detta ska göras utan tillgång till den fysiska utrustningen.

Dessa mål kan delas upp i två delar. Den största av dem är genomförandet av beredningen med hjälp av Process Designer och Process Simulate (PD/PS). Den andra delen är att göra en täckande efterforskning kring ämnet för att se vilka fördelar och nackdelar det finns med metoderna som ska utvärderas.

I den virtuella modellen ska montering och ihopsvetsning av tio delar som utgör en front till en Volvo S80 simuleras. Simuleringen ska vara händelsebaserad. In- och utmatning av delarna skall även finnas med i simuleringen.

Verktygen som ska användas är Process Designer och Process Simulate från Siemens [8][9]. Till modellen ska det skapas ett PLC-program, vilket kommer att ske i en tidig beredningsfas. Styrprogrammet ska verifieras och valideras mot den skapade modellen.

Den virtuella modellen ska följa Volvo Cars standard för programmering av robotceller. Hela logikuppbyggnaden skall följa standarden.

1.6 Avgränsningar

De virtuella modeller som kommer att användas är de befintliga modellerna över PSL som finns skapade i tidigare projekt. Nya virtuella modeller som andra kandidatgrupper skapar, eller tidigare skapat, kommer även de att användas. Enbart de delar som inte går att tillgå från varken CTH eller VCC, kommer modelleras upp. Dessa modeller kommer vara av enklare karaktär, utan hållfasthetsberäkningar.

Att köra produktionscellen i verkligheten med hjälp av PD/PS kommer inte att utredas. All kommunikation med en OPC-server lämnas till senare projekt.

In- och utflöde av material begränsas till lokalens uppbyggnad. Placering av material för intransport samt uttransport av färdig front antas göras manuellt.

1.7 Metod

För att skapa en fungerande simulering krävs arbete i både Process Designer och Process Simulate. Dessa två programvaror är separata men beroende av varandra.

I PD skapas strukturen kring processen. Detta innefattar definiering av stationer, materialflöden och vilka resurser som används i vilken del av processen. Även de ingående detaljerna knyts till operationer inom stationerna.

När en struktur är skapad i PD exporteras denna till PS där simuleringen byggs upp och körs. Operationerna utformas och logiken skapas. Här kontrolleras även kollision, räckvidd och cykeltider för processen.

Befintliga virtuella modeller från andra aktuella kandidatprojekt samt från Chalmers server med modeller över PSL används. Vid behov skapas nya modeller av verklig utrustning i CATIA. Genom att sätta ihop flera delar (parts) och ge dem förhållningssätt (constrains) till varandra går det att skapa produkter med möjlighet till rörelser.

I och med att CATIA och PD/PS inte är av samma tillverkare måste dessa färdiga produkter konverteras till kompatibelt format och PD/PS använder sig av JT-formatet. Vid konverteringen överförs produktens egenskaper till det nya formatet. Förhållningssätt mellan delarna i modellen, material, färg och självklart mått överförs alla oförändrat. När de överförda filerna är konverterade till JT-format går det fortfarande att ändra dess attribut, dvs. det går fortfarande att ändra dess egenskaper och går således att fortsätta att arbeta med.

Händelsestyrd simulering kräver sensorer. Vid definiering av en sensor skapas eller flyttas en geometri in i PS. Denna geometri går att definiera som en sensor och kommer då inte vara en fysisk del i en sensor utan geometrin är det sensorn känner av. Exempelvis definieras att sensorn ska ge utslag om någon given produkt är närmare än ett visst givet mått den geometri som nyss definierats. Den fysiska delen av sensorn modelleras upp, men är inget “smart” objekt, utan den är enbart en geometri utan funktioner.

2

Teori

I följande kapitel beskrivs den teori som behövs för att förstå de metoder och tillvägagångssätt som används för att kunna undersöka virtuell logikberedning och virtual commissioning som arbetsmetod. Det innefattar teori kring simulering, programvaror, robotteori samt logikuppbyggnad. Även teori bakom begreppet virtual commissioning tas upp.

2.1 Simulering

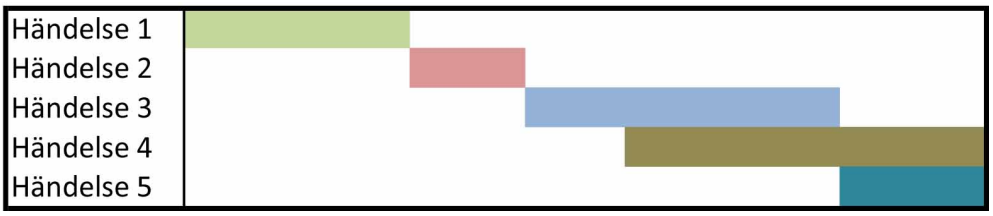
"Att representera ett system med ett annat i avsikt att studera dess dynamiska uppförande eller för att under laboratorieförhållanden träna behärskandet av systemet." Citat från Nationalencyklopedin [10]

Med simuleringsverktyg går det att experimentera och undersöka "vad händer om"-situationer utan att störa pågående produktion och utan att riskera att maskiner tar skada. Det är även möjligt att göra dessa experiment utan att ha tillgång till några fysiska maskiner.[3]

Det finns två olika tillvägagångssätt att arbeta utefter när man ska ta fram en simulering, tidsstyrda- och händelsestyrda simuleringar.

2.1.1 Tidsstyrd simulering

Vid tidsstyrd simulering sätts villkor upp för vad som skall hända genom att definiera vid vilken tidpunkt händelsen skall ske. Variablerna som styr simuleringen bestäms alltså kontinuerligt med tiden. En grafisk representation av tidsstyrd simulering är ett Gantt-schema, där händelser är kopplade efter varandra med olika långa staplar beroende på tidsåtgång.[11] Se figur 2.1.



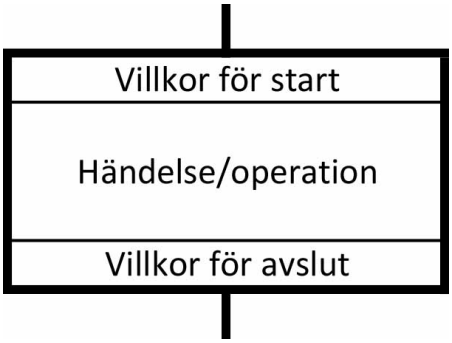
Figur 2.1: Exempel på ett Gantt-schema

Vid en tidsstyrd simulering sker händelserna efter varandra och kommer att ske i samma ordning varje gång oavsett vad som händer.

En nackdel med tidsstyrd simulering är att ingen hänsyn tas till om något oväntat inträffar. Däremot ger det stor noggrannhet och kontroll på händelserna samt i vilken ordning och skede händelserna sker.[11]

2.1.2 Händelsestyrd simulering

Vid en händelsestyrd simulering utförs en operation när dess specifika uppsättning av villkor är uppfyllda, se figur 2.2. Detta sätt att styra innebär att en händelse inte startar utefter en förutbestämd ordning, utan när startvillkoret är uppfyllt. Då styrningen i produktion sköts på detta sätt blir en simulering mer verklighetstrogen än om den varit tidsstyrt.[12]



Figur 2.2: Exempel på villkor för en operation

En av de komponenter som skiljer en händelsestyrd från en tidstyrd simulering är utnyttjandet av givare.

PD/PS använder sig av två typer av givare, närvarogivare och ledpositionsgivare.
Närvarogivaren ger signal då ett objekt är i närheten eller i kontakt med sensorn. En närvarogivare måste vara grafiskt representerad i PS för att fungera.
Ledpositionsgivaren ger ett värde beroende på vilken position en led befinner sig i. Detta värde kan tolkas till en given position. Denna givare behöver inte ha en grafisk representation i PS för att fungera.

2.2 Virtuell produktionsberedning

Virtuell beredning av ett produtktionssystem innebär att produktionen modelleras upp virtuellt. Det kan vara från att enbart använda sig av enskilda modeller av de komponenter som ingår till att modellera upp alla geometrier och verktyg i en produktionscell. Det är alltså möjligt att skapa modeller av hela fabriker om det skulle behövas. Virtuell beredning möjligör kontroll av utrymmet i produktionen samt att all utrustning och alla produktdelar fungerar ihop i processen.

2.3 Virtuell beredning av logik

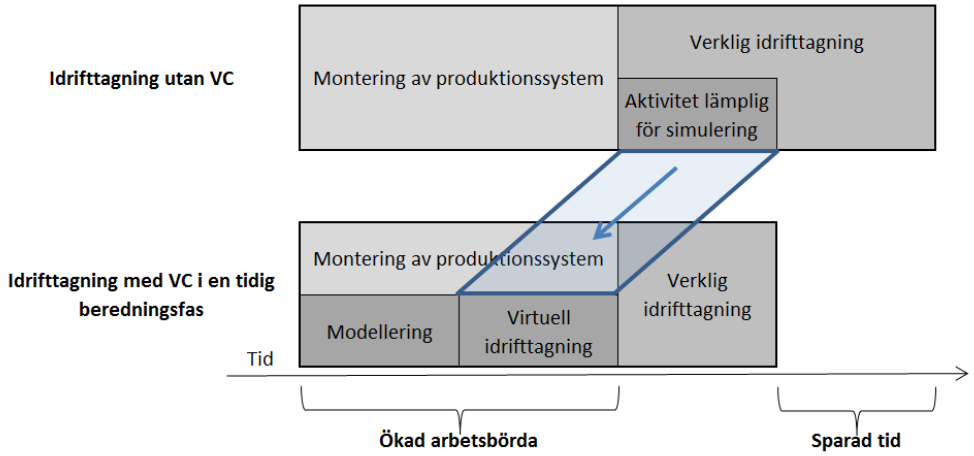
Utifrån den virtuella modellen som byggs upp skapas programkod. Det är möjligt att bereda denna logik på flera sätt, men inom virtuell beredning görs detta virtuellt i datorn. Denna programkod kan sedan användas vidare i simuleringar eller användas direkt i verkliga produktionssystemet.

2.4 Virtual Commissioning

Virtual Commissioning (VC) innebär att en programkod som tidigare tagits fram körs i den virtuella modellen. En realistisk simulering görs för att validera styrsystemet för produktionsutrustning före implementering i verkligheten. VC ger ingenjörer nya möjligheter att designa komplext beteende och förbättringar av prestanda i adaptiva tillverkningssystem.[7]

Idag uppskattas det att programmeringsingenjörerna bidrar med 60% av funktionaliteten i en högt automatiserad utrustning. Denna siffra uppskattas öka med tiden. [1]

Idrifttagningsfasen i ett projekt kan vara upp till en fjärdedel av den totala projektcykeltiden, varav 90% av denna tid används till förseningar och aktiviteter relaterade till elektrisk kontrollutrustning. Samtidigt används 70% av dessa 90% till att rätta till fel i programvaran. Det vill säga att det läggs ner 60% av programmeringstiden på att leta och rätta till fel, och det motsvarar upp till 15% av hela time-to-market. Tiden det tar att implementera och göra om program i produktionssystem vid verklig idrifttagning går att korta ned med hjälp av VC.[1] Se figur 2.3



Figur 2.3: Besparing av tid med VC i tidig beredningsfas

2.4.1 Hur fungerar VC?

Virtual commissioning ger möjlighet till besparing av tid och minskade kostnader genom att programmeringen tar över initiativet vid design av system. Detta ger en möjlighet att i ett tidigare stadie kunna påverka designen av produktionsverktygen.[1]

Följande faktorer behövs enligt Drath, Weber och Mauler [2] för en framgångsrik introduktion till VC i industrin:
Riktig kontrollkod: Koden för PLC, robotar och HMI skall kunna användas oförändrad oavsett åt vilket håll som den överförs och bäddas in.
Riktiga ingenjörsverktyg: Existerande ingenjörsverktyg och språk för PLC-, robot- och HMI-applikationer måste bevaras. Detta för att kunna använda äldre, redan existerande bibliotek och standardlösningar.
Inbäddningsbarhet: Den virtuella miljön skall lätt kunna bäddas in i redan existerande arbetsflöden.
Utökningsbarhet: Den måste stödja flera olika PLC-system, då det finns många PLC-märken på marknaden.
Virtuella robotkontroller: Beteendet på det framtagna programmet skall kunna köras på den virtuella kontrollen och bete sig identiskt med den verkliga robotkontrollen.

2.4.2 Nackdelar med VC

När beslutet om att använda VC tas måste en medvetenhet om att investeringskostnaderna är höga, vilket kan medföra en risk för företaget. Det kan vara kostnader för ingenjörskunskaper inom området i form av utbilning och kompetens samt för programvaror.

Samtidigt kan en minskad projektkostnad vara dold för ingenjörerna som ofta bara ser den ökade arbetsbördan. Besparingar sker i andra delar av processen som ingenjörerna vanligtvis inte ser. Dessutom betalar inte kunderna för simuleringar utan för ett levererat och fungerande produktionssystem. Kunderna ser inte simuleringarna och arbetet bakom.[1]

Industrin har varit motvillig att införa VC på grund av dess nackdelar med hög komplexitet, kostnad under introduktionen och variation av leverantörer samt att metoden kräver hög kunskapsnivå.[2]

2.5 Programvaror

De programvaror som använts vid utvärderingen av virtuell beredning av programlogik och virtual commissioning är CATIA, Process Designer och Process Simulate. DELMIA är ett alternativ till PD/PS, i vilka det är möjligt att simulera produktionssystem. De är så kallade Product Lifecycle Management-program (PLM).

2.5.1 CATIA

CATIA är en programvara från Dassault Systèmes och är ett kraftfullt CAD-program för tredimensionella modelleringar och konstruktionslösningar. I programmet kan samband mellan delarna och funktioner i modellerna kopplas.[13] Dessa filer kan sedan konverteras, med bibehållen geometri och funktion, till JT-format som används av PD/PS.

2.5.2 DELMIA

Programvaran DELMIA är från Dassault Systèmes och kan beskrivas som en motsvarighet till PD/PS. Det används för att virtuellt skapa, planera och kontrollera produktionsprocesser i tre dimensioner. Med hjälp av simuleringarna är det möjligt att optimera fabriker och produktionssystem. Programvaran fungerar som en bro mellan den virtuella världen med CATIA-modeller och den verkliga världen.[14]

2.5.3 Process Designer

Process Designer (PD) är tillsammans med Process Simulate (PS) två programvaror, som ingår i programserien Tecnomatix, och är framtagna av Siemens. Med en uppsättning av driftkompatibla verktyg kan produktionssystem designas och modelleras med programvaran. Hjälpmedel som finns att tillgå är Gantt- och PERT-diagram samt scheman och tabellvyer. Programvaran kommer med funktioner för ändringshantering och analys av tillverkningsegenskaper, resurser, drift och varianter i tillverkningen. I programmet finns möjlighet till beräkning av kostnader och tidsåtgång samt linebalansering och användning av spårningsverktyg.[8]

2.5.4 Process Simulate

Med programvaran Process Simulate görs simuleringar i 3D av produktionssystem, detektering av både statiska och dynamiska kollisioner, ihopsättning och planering av robotbanor. Det används till att designa och simulera produktionslinor och arbetsstationer samt att virtuellt idriftsätta dem. Det finns även möjlighet att simulera människors arbete. PS stödjer både tids- och händelsestyrda simuleringar.[9]

2.6 JT

Filformatet JT är framtaget av Siemens och är sedan 2012 en accepterad ISO-standard för att visa och dela filer med 3D-produktinformation. Dessa filer kan användas direkt av CAD och PLM-programvaror.[15] Idag är dataformatet utbrett och används i bil- och flygplansindustrin [16].

2.7 Robotteori

Vid robotprogrammering, både virtuellt och fysiskt, placeras punkter ut i 3D-rymden dit robotens huvud skall röra sig. Hur robotens huvud färdas mellan två förbestämda punkter kan ske på olika sätt. Färdbanan huvudet färdas varierar mellan inställningarna. Exempelvis kan färdbanan vara en rät linje eller en cirkulär båge mellan punkterna. Den kan även vara av typen att lederna på roboten skall röra sig så lite som möjligt.

Det finns flera alternativ kring hur roboten beräknar färdvägen med hänsyn till hur de olika lederna på roboten rör sig. Den biten av programvaran som styr dessa beräkningar kallas för controller. Varje tillverkare av robotar har utvecklat egna controllers för att styra sina specifika robotar, och dessa finns representerade som mjukvaruinstick till PS. PS har en egen controller, men denna beräknar inte ledernas rörelse enligt samma metodik som tillverkarnas. För att simuleringen ska bli så verklighetstrogen som möjligt krävs därför mjukvaruinsticken.

2.8 Logikuppbyggnad Volvo Cars

När arbete ska utföras i PD/PS är det viktigt att det finns en given struktur att följa. Om det finns brister i denna struktur är det lätt att simuleringens funktioners missas, vilket kan medföra att en fullgod simulering ej kan genomföras.[17]

Volvos standarder gällande funktionsblock har två aspekter att ta hänsyn till. Den första är hur man bygger upp signalerna och den andra är de logiska blocken som används.[17]

2.8.1 Signaler

Det finns totalt åtta stycken standardsignaler som VCC använder sig av. Dessa är framtagna för att signalhanteringen i simuleringen ska vara så nära anknuten till hur det ser ut i verkligheten som möjligt.[17]

LockNumIn: En Interlockinsignal som exempelvis används för att se om en robot kan gå in i ett specifikt utrymme eller inte.
LockNumOut: En Interlockutsignal som används för att begära inträde i ett specifikt utrymme.
SmallDress: Beordrar roboten att utföra en SmallDress.
BigDress: Beordrar roboten att utföra en BigDress.
CycleDone: Informerar PLCt att en arbetscykel har utförts.
DressDone: Informerar PLCt när roboten har utfört en "Dressing".
DirectionIn: En CASE-signal som informerar roboten vilken uppgift som ska utföras.
DirectionOut: En CASE-signal som används av roboten för att fråga PLC vilka uppgifter som ska göras.

2.8.2 Signaloperationer

WaitSignal:
Operationen används av en robot då den vill gå in i ett specifikt utrymme, exempelvis en zon eller station. Operationen skickar ett förbestämt värde på LockNumOut och väntar på ett likadant svarsvärde på LockNumIn. Först när samma värde skickas tillbaka kan roboten gå in i utrymmet. När roboten är klar med sin uppgift skickas en signal tillbaka på LockNumOut med föregående värde + 1.[17]

Exempel på detta:
1,3,5. . . : Station1, Station2, Station3 är upptagen.
2,4,6. . . ;Station1, Station2, Station3 är ledig.
Samma standard gäller även för zoner.

WaitCase:
Operationen används då roboten ska göra ett logisk val mellan uppgifter. Om en robot har i uppgift att först ladda en fixtur med delar som sedan ska svetsas ihop konsulteras operationen. Om roboten ska fylla fixtur eller svetsa beror på om det redan ligger en del laddad. WaitCase kommer att skicka en signal på DirectionOut och sedan vänta på ett svar kring vilken uppgift som ska genomföras. Roboten kommer att få en signal på DirectionIn med ett förbestämt värde som den tolkar för att se vilken uppgift som ska utföras.[17]

2.8.3 Logikblock

Allocate_Zone:
Blocket håller reda på vilka zoner i en station som är allokerade, upptagna, och vilka som är lediga. Blocket får sina signaler från WaitSignal-operationen. När en förfrågan kring en zon uppstår kollar Allocate_Zone huruvida zonen i fråga är ledig. När denna blir ledig kommer en signal skickas tillbaka med den efterfrågade zonens status. För att göra detta används signalerna LockNumOut och LockNumIn för att kommunicera med simuleringen.[17]

Robot_StartBlock:
Blocket används för att ge roboten kommandon om att starta. Blocket skickar en start-signal till roboten då den har slutfört sin arbetscykel, vilket gör att robotprogrammet loopas.[17]

ToolChange:
Blocket innehåller instruktionerna för att utföra ett verktygsbyte mellan två verktyg. Blocket är skapad som en grupp operationer, en compound, i simuleringsmodellen. Sekvensen kräver att det finns standardiserade platser för att fungera. Dessa platser måste vara namngivna så att given robot kan utnyttja dessa. Denna namngivning ska följa robotmärkets standard så att roboten vet i vilken ordning de ska utföras.[17]

Operation	**Innehållsbeskrivning**
HomeToStand	Förflyttar roboten från hemmapositionen för verktyget till verktygshållaren.
StandToPutChk	Förflyttar roboten till en position precis ovanför verktygshållaren.
PutChkToBetw	Förflyttar roboten från tidigare position till hemmapositionen för roboten.
BetwToStand	Förflyttar roboten från hemmapositionen till verktygshållaren.
StandToGetChk	Förflyttar roboten från verktygshållaren till en position precis ovanför.
GetChkToHome	Förflyttar roboten från tidigare position till hemmapositionen för verktyget.

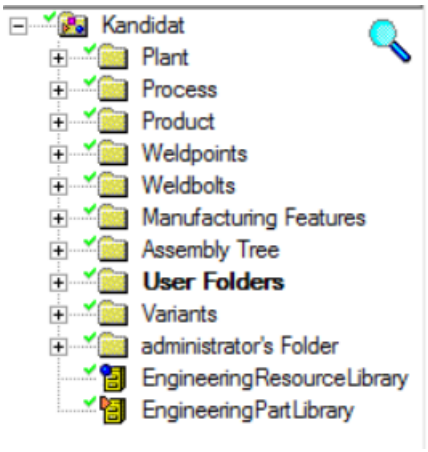
Tabell 2.1: Robotoperationer för verktygsbyte

Beroende på huruvida ett verktyg ska lämnas tillbaka eller hämtas upp kan sekvensen köras i olika ordning.

2.9 Volvo Cars ramverk för trädstruktur

I PD/PS finns ingen given trädstruktur. För att arbetet i programvaran ska gå så smidigt som möjligt krävs en organiserad struktur. Därför har VCC tagit fram en standardiserad trädstruktur som ska följas. I tabellerna nedan beskrivs denna standardisering från en hög till en låg nivå.

Tabell 2.2 beskriver den högsta trädstrukturen, Main tree. Alla noder utom -Library är mappar som innehåller respektive noder som beskriver processen på angivet område. Se figur 2.4 för motsvarande struktur i PD/PS.

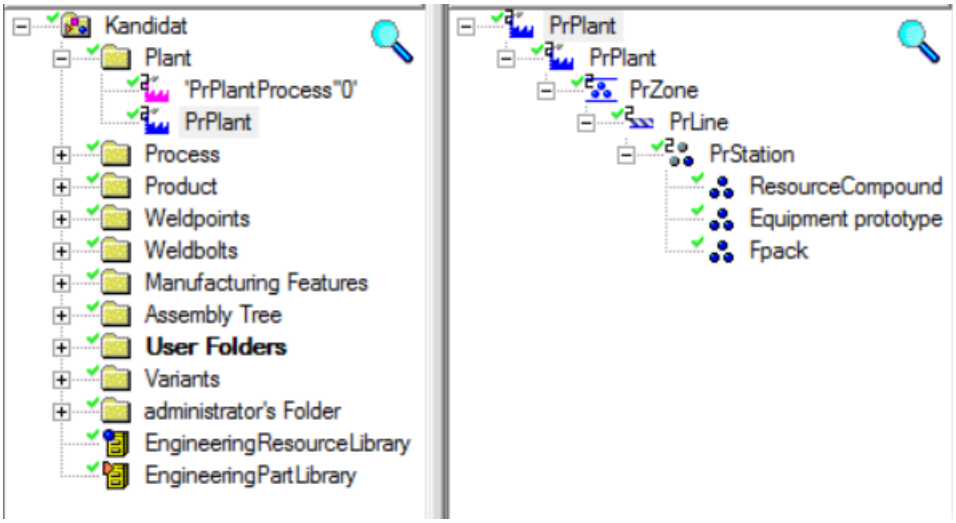


Figur 2.4: Main tree i PD/PS

Namn	**Innehållsbeskrivning**	**Klass**
Plant	Den övergripande fabriken.	Collection
Process	Innehåller tillverkningsprogrammet och dess processer som ligger under respektive Plant.	Collection
Product	Innehåller alla delar som är relevanta för respektive process.	Collection
Weldpoints	Innehåller alla svetspunkter som är relevanta för de processer som körs.	Collection
Weldbolts	Innehåller alla bultförband som är relevanta för de processer som körs.	Collection
Continous features	Innehåller diverse egenskaper och data relevanta för processerna i respektive Plant. Beskrivs nedan som Manufacturing Features.	Collection
Assembly tree	Innehåller alla noder som programmet genererar relaterade till In Process Assembly(IPA).	Collection
User Folders	Användarens mapp.	Collection
Variants		Collection
Studyfolders	Innehåller alla Studies som används vid simuleringar, svetslayouts och dyl.	Collection
EngineeringResourceLibrary	Generell mapp innehållande Resource Library.	RobcadResourceLibrary
EngineeringPartLibrary	Innehåller alla importerade Parts.	RobcadPartLibrary

Tabell 2.2: Övergripande trädstruktur, Main tree

Under Plantmappen finns följande struktur, se figur 2.5. Plant beskriver hur miljön som simuleringen ska utföras i ser ut. Noden är uppbyggd av komponenter från noderna i tabell 2.2. Plant ska innehålla robotar, fixturer verktyg etc. I tabell 2.3 beskrivs alla klasser i Plant.

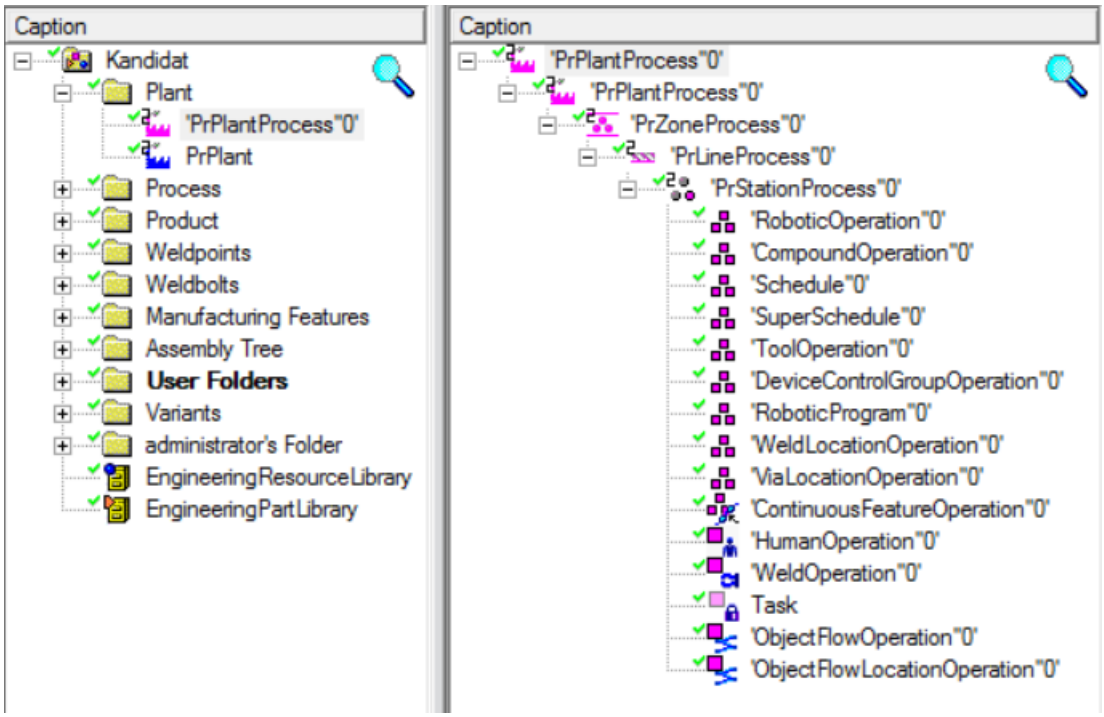


Figur 2.5: Plant tree i PD/PS

Plant	**Innehållsbeskrivning**
	Nedanstående klasser är i fallande ordning i trädhierakin.
Site	Vilken ort fabriken befinner sig (Torslanda exempelvis).
Plant	Anger vilken fabrik på orten.
Workarea	Används för att gruppera Resources som används inom ett Plant.
Line	En produktionslina.
Station	Station längst produktionslinan.
ResourceCompound	Används för att gruppera ResourceObjects.
Equipment prototypes	Används för att gruppera utrustning i exempelvis en fixtur eller greppverktyg.
Fpack	Används för att gruppera utrustning relaterad till en robot.

Tabell 2.3: Trädstruktur för Plant, Plant tree

Tabell 2.4 beskriver tillverkningsprocessen som ställs upp i PD. Här beskrivs operationerna och deras plats i heirarkin. Under operationerna ligger information kring hur de ska utföras, exempelvis färdbanan för ett robotprogram, se figur 2.6.

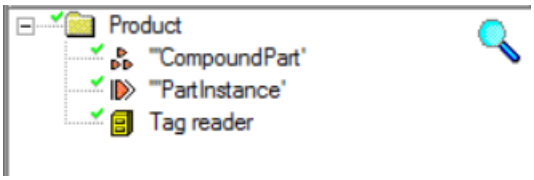


Figur 2.6: Process tree i PD/PS

Process	**Innehållsbeskrivning**
PlantProcess	Topnoden i tillverkningsprocessens program.
AreaProcess	Använd då inget direkt maskinnummer finns att relatera till en grupp Processes.
LineProcess	Används för att gruppera stationer.
StationProcess	Grupperar alla Resources so används i en station.
RoboticOperation	Grupperar operationer som utförs av en robot.
CompoundOperation	Grupperar exempelvis robotoperationer.
Schedule	Grupperar exempelvis robotoperationer.
SuperSchedule	Grupperar exempelvis robotoperationer men innehåller även de vanliga segmenten för en robot.
ToolOperation	Grupperar fixturoperationer.
DeviceControlGroupOperation	
WeldOperation	Beskriver svetsbanor.
RoboticProgram	Används för att ladda upp och ner operations och schedules för en robot.
WeldLocationOperation	Svetspunkter
ViaLocationOperation	
ObjectFlowOperation	
ObjectFlowLocationOperation	
Task	Används vid simulering.
ContinuousFeatureOperation	
HumanOperation	Används för att gruppera människor vid en station.

Tabell 2.4: Trädstruktur för Process, Process tree

Productmappen, som beskrivs i tabell 2.5, agerar som ett bibliotek för de detaljer som kommer att användas i processen. Efter att detaljerna lyfts in i PD/PS placeras de i denna nod, se figur 2.7.

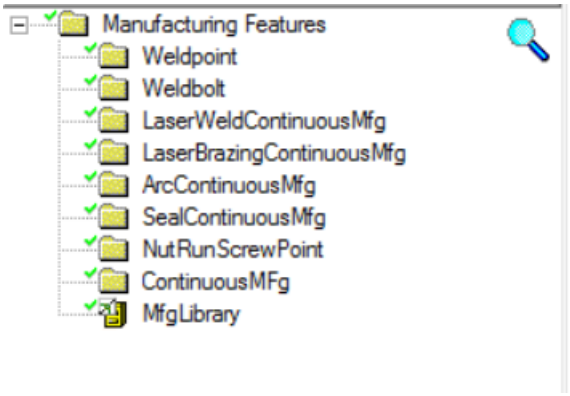


Figur 2.7: Product tree i PD/PS

Product	**Innehållsbeskrivning**
PmCompoundPart	
PmPartinstance	
Tag reader	

Tabell 2.5: Trädstruktur för Product, Product tree

Tabell 2.6 beskriver strukturen i Manufacturing Features. Här sparas all data kring svetspunkter, limsträngar och liknande som beskriver hur tillverkningsprocesser ska gå till. Figur 2.8 visar Manufacturing Features i PD/PS.



Figur 2.8: Manufacturing Features i PD/PS

Manufacturing features	**Innehållsbeskrivning**
MfgLibrary	Är topnoden för alla Mfg_Features under given plats.
Weldpoint	
Weldbolt	
LaserWeldContinuousMfg	
LaserBrazingContinuousMfg	
ArcContinuousMfg	
SealContinuousMfg	
NutRunScrewPoint	
ContinuousMfg	

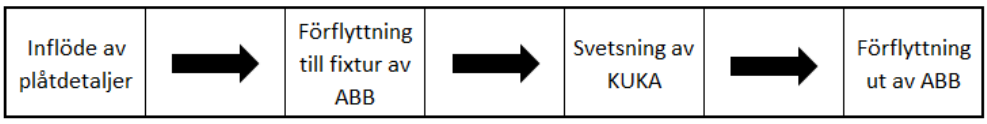
Tabell 2.6: Trädstruktur för Manufacturing features

3

Genomförande

För att en virtuell beredningsprocess skulle bli fullständig fanns det flertalet moment som var tvunget att genomföras. I följande kapitel beskrivs de moment för att göra utvärderingen möjlig. Vissa av dessa moment har arbetats med parallellt.

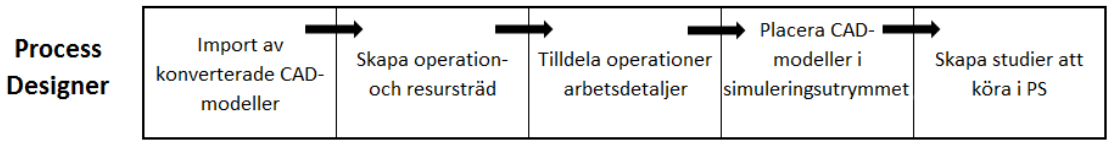
Produktionssystemet som simulerades använder sig av de två största robotarna i produktionscellen i PSL, ABB IRB 4600 och KUKA 30-3. Produktflödet är enligt figur 3.1. Plåtdelajer transporteras in till robotarnas arbetsområde. Den största biten läggs på ett bord medan de nio mindre detaljerna transporteras in på transportbandet. När detaljerna är framme plockar ABB-roboten upp smådetaljerna med gripverktyget, två åt gången och placerar dem i fixturen. Den största biten är så stor att den måste lyftas in ensam. När alla plåtdetaljer är på plats tar KUKA-roboten över och går in och punktsvetsar samman alla detaljer. Efter att KUKA-roboten är klar lyfter ABB-roboten ut den färdiga frontdetaljen ur cellen.



Figur 3.1: Blockschema över produktionscellens arbetsgång.

Arbetsgången i Process Designer går att följas i figur 3.2. Det börjar med att befintliga, redan skapade CAD-modeller av alla relevanta delar importeras. Formatet på dessa modeller måste vara av JT-format. Modellerna behöver vara allt som behövs för att återspegla verkligheten i produktionssystemet t.ex. golv och väggar, robotar, material och robotskåp. Mängden modeller och detaljrikedomen kan göras hög, men enbart relevanta delar bör föras in. När modellerna är importerade är nästa steg att skapa operations- och resursträd i trädstrukturen. Arbetsdetaljer tilldelas operationerna. Detta för att definiera

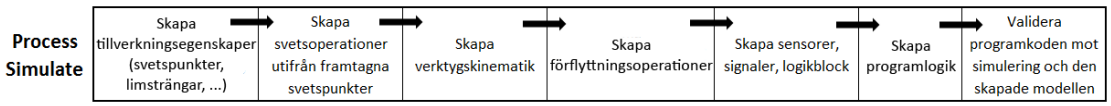
vilka detaljer som hör till operationerna, så att de går att manipulera i ett senare skede av simulerningen. Nästa steg är att placera ut CAD-modellerna i simuleringsutrymmet. Tidigare har de enbart existerat i objekträd och inte varit representerade grafiskt. Placeringarna utgår ifrån ett koordinatsystem i simuleringsutrymmet och placeras ut som de skulle ha placerats i verkligheten. Till sist skapas studies, studier.



Figur 3.2: Blockschema över arbetsgången i PD

Efter att studies har skapats kan dessa köras i Process Simulate. Arbetsgången i Process Simulate går att följas i figur 3.3. När en studie är öppnad i Process Simulate är det möjligt att tilldela komponenterna tillverkningsegenskaper så som svetspunkter och limsträngar m.m. Med hjälp av dessa egenskaper definieras hur tillverkningen skall gå till. För att sedan få modellerna att röra på sig och bete sig som en riktig cell skulle göra skapas tillverkningsoperationer. I detta fall skapas svetsoperationer, då detaljerna skall svetsas ihop. När operationerna är klara måste vertygens rörelser definieras. Verktygskinematiken bestämmer hur verktygen skall förflytta och rör sig, exempelvis mellan två svetspunkter.

För att materialet skall kunna följa med flödet som finns genom en produktion behöver förflyttningsoperationer skapas. Dessa operationer talar om hur material tas in, förflyttas, för att sedan tas ut ur cellen och simuleringen. När det är bestämt hur materialet skall bearbetas och förflyttas måste villkor för när detta skall ske definieras. Sensorer, signaler och logikblock skapas för att bestämma detta. Sensorer känner av när material är placerade rätt och använder sig av signaler kopplade till dessa för att sedan aktivera logikblock av flera signaler. Dessa logikblock kopplas sedan samman för att skapa en hel programlogik. Det sista steget i denna simuleringsprocess är att simulera de modeller som skapats och programmerats för att verifiera programkoden mot den virtuella modellen.

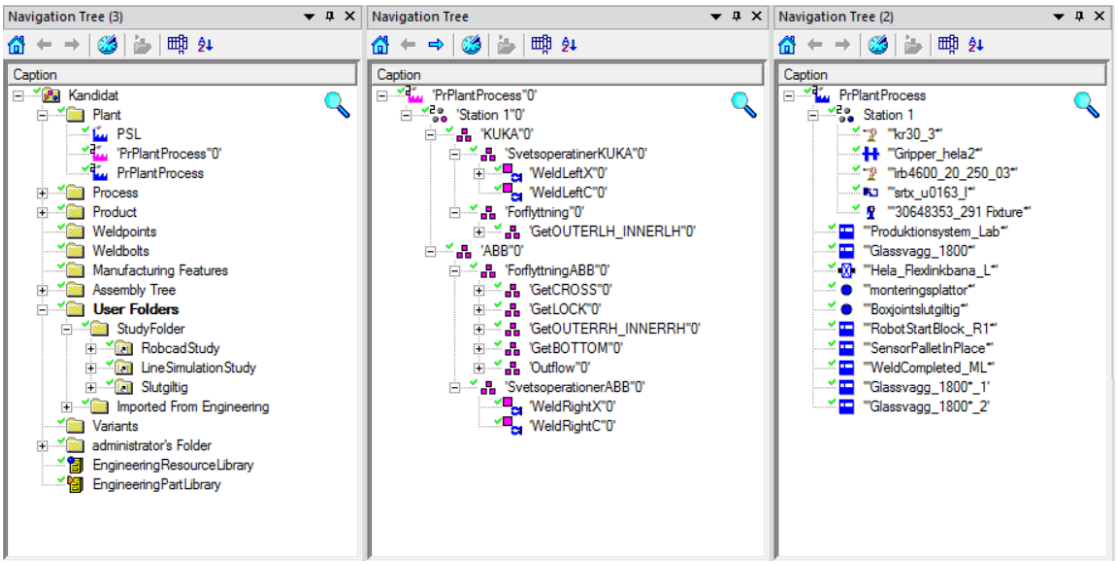


Figur 3.3: Blockschema över arbetsgången i PS

Arbetet i Process Designer och Process Simulate är inte låsta utan det går att göra moment både tidigare och senare än den beskrivna arbetsgången. Det är även möjligt att gå tillbaka och ändra, ta bort och lägga till saker i studier i Process Designer, trots att arbete har påbörjats i Process Simulate.

3.1 Trädstruktur

För att arbetet med PD och PS skulle kunna genomföras krävdes en organiserad struktur i det objektträd som arbetet utfördes i. Här krävdes en utarbetad standard och struktur som medförde att andra personer inte skulle behöva vara insatt i det specifika programmet för att kunna navigera i trädet. Se figur 3.4. Trädstrukturen som användes följer Volvos Cars ramverk för trädstruktur i PD/PS.



Figur 3.4: Exempel på objektträd i PS

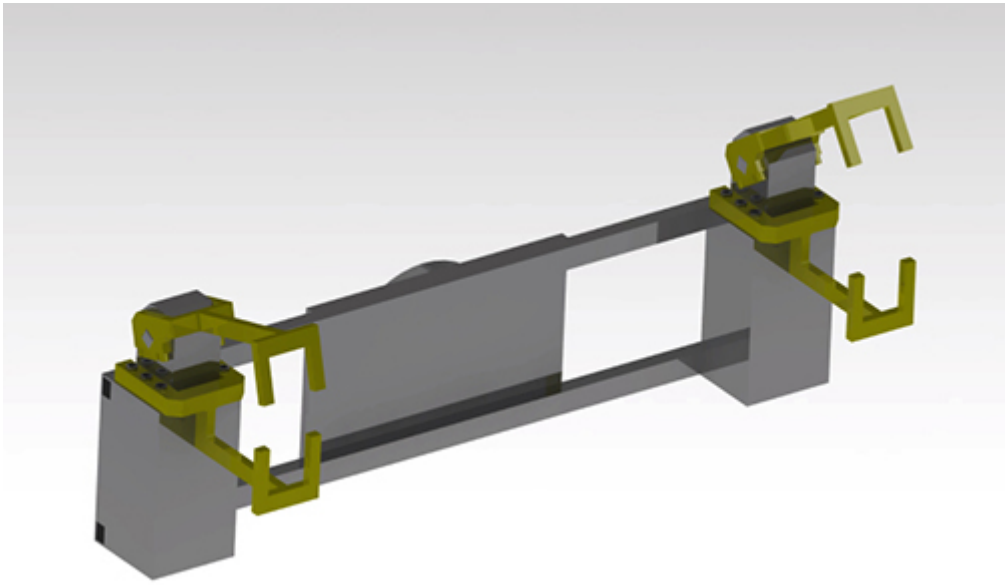
3.2 CAD-Modeller

CAD-modeller användes för att bygga upp hela den virtuella modellen av produktionscellen och dess innehåll i PD/PS. Det fanns tidigare projektarbeten som hade skapat CAD-modeller av produktionscellen som kunde utnyttjas. En inventering av en server på Chalmers med modeller över PSL krävdes för att skapa en uppfattning kring vad som fanns och vad som fattades. Vidare krävdes att användbara CAD-filer konverterades till JT-format för att bli brukningsbara i PD/PS.

3.2.1 Skapande av eget gripverktyg

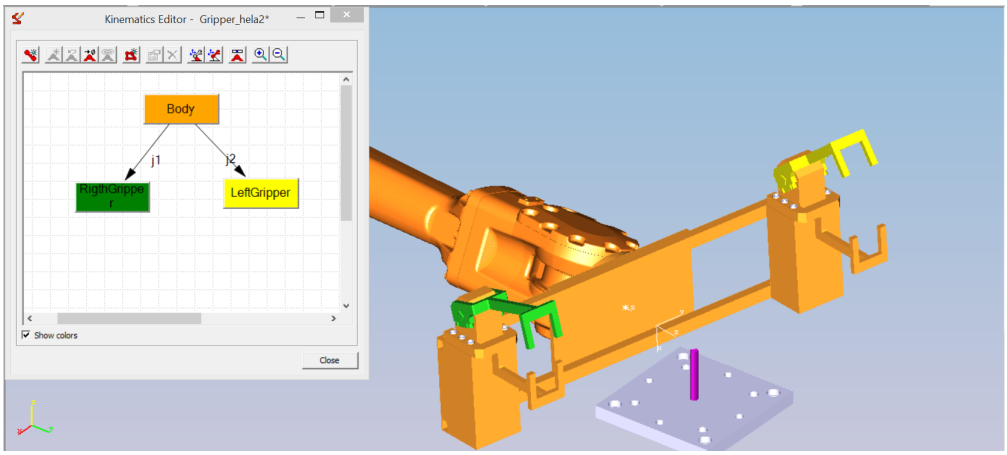
Gripverktyget, se figur 3.5, designades så att den har två klamrar för att kunna ta en bit med varje klammer och den stora med båda. Momenten som gripverkyget skall utföra är att först hämta den största frontdelen och sätta den på plats i fixturen. Sedan hämtas övriga delar, två delar i taget, och sätts i fixturen. Alla detaljer som skulle hanteras av gripverktyget var inte helt, men nästan plana. Detta medförde att gripverktyget inte krävde några komplicerade geometrier för att kunna greppa alla detaljer och att samma

verktyg kunde användas i alla förflyttningsoperationer. Anledningen till att gripverktyg med sugkoppar inte användes är efter rekommendationer från Andreas på VCC att enbart använda sugkoppar vid större slätare delar.



Figur 3.5: Det skapade gripverktyget

I PS definierades gripverktygets rörelsemönster. Gripverktygets leder och gripande komponenter bestämdes och dess rörelsevidd sattes tillsammans med de olika positionerna gripverktyget kan stå i. Detta gjordes i en miljö som visas i figur 3.6.

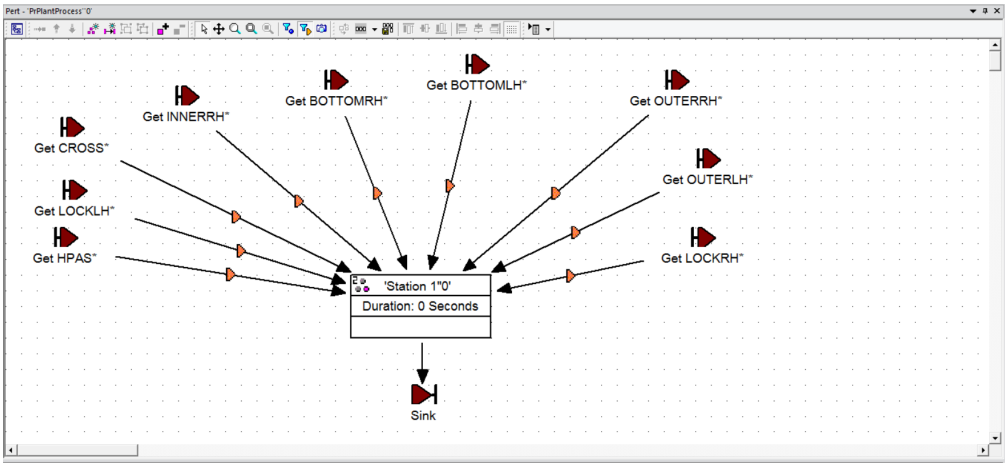


Figur 3.6: Definiering av gripverktygets rörelsemönster

3.3 Materialflöde

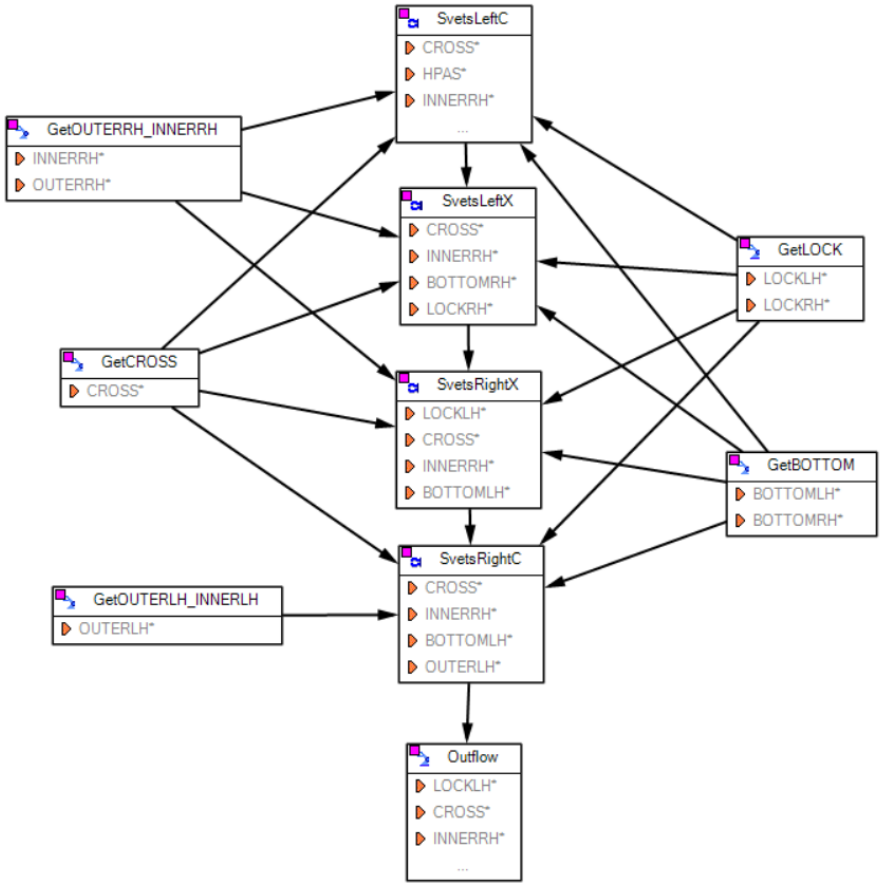
Genom produktionscellen sker ett materialflöde. I och med att produktionscellen inte var anpassad för denna typ av tillverkning krävdes ett nytt materialflöde. Den största frontdelen var för stor för den befintliga transportbanan och hämtas därför in från ett bord vid sidan av. De resterande nio delarna transporteras in i cellen på transportbanan. Den färdiga fronten transporteras ut genom att den läggs på samma bord som den största frontdelen hämtas från.

I PD/PS krävdes det att materialflödet definierades i en flödesstruktur via verktyget *PERT-viewer*. *PERT-viewer* visar materialflödet på en hög nivå och beskriver hur komponenter konsumeras på en stationsbas. Se figur 3.7. Eftersom produktionscellen i PSL enbart bestod av en station medförde det ett mycket enkelt materialflöde på denna nivå.



Figur 3.7: Materialflöde i *PERT-viewer*

Ett annat sett att visualisera materialflödet var med funktionen *Material Flow Viewer*. Detta schema fyller samma funktion som *PERT* men på en mer detaljerad nivå. I *Material Flow Viewer* beskrivs materialflödet på en station, se figur 3.8. Stationen i produktionscellen var uppbyggd av flera operationer som i sin tur konsumerade och genererade komponenter. På denna nivå blev flödet mer komplext, men fortfarande överskådligt.



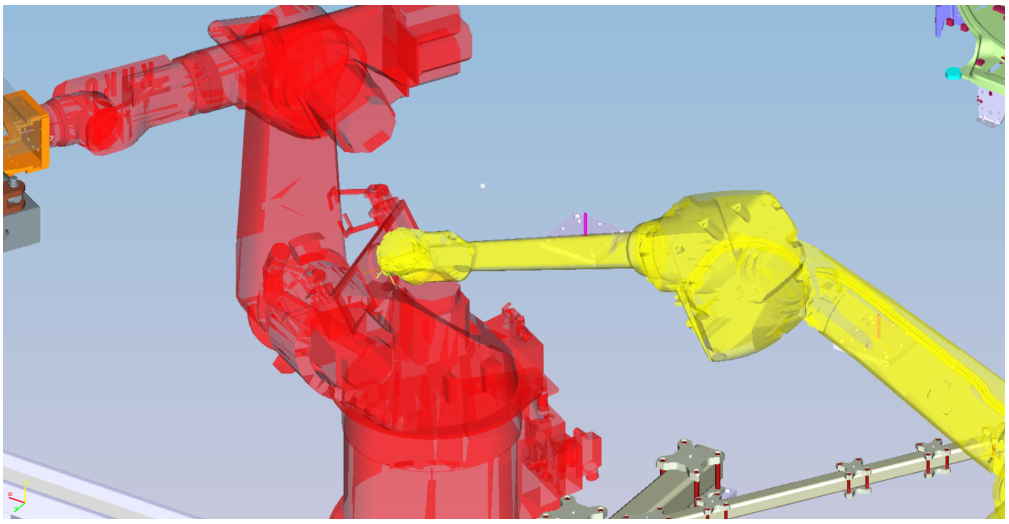
Figur 3.8: Materialflöde i Material Flow Viewer

3.4 Kinematik

En central del i framtagandet av en simulering var skapandet av kinematiken för material, detaljer och maskiner i produktionscellen. Här var hänsyn tvungen att tas till hur materialflödet i cellen såg ut. Någon form av optimering kring exempelvis robotarnas rörelser var även tvunget att finnas, exempelvis ta fram kortaste eller effektivaste transportvägarna för material. Stor hänsyn togs även kring zonallokering för att hindra kollisioner.

Kollision: För att upptäcka kollisioner vid exempelvis förflyttning av material finns verktyget *Collision Viewer* i PS. När en sekvens simulerades i programmet var det enkelt att samtidigt köra en kollisionskontroll. Innan kontrollen skedde specificeras kom-

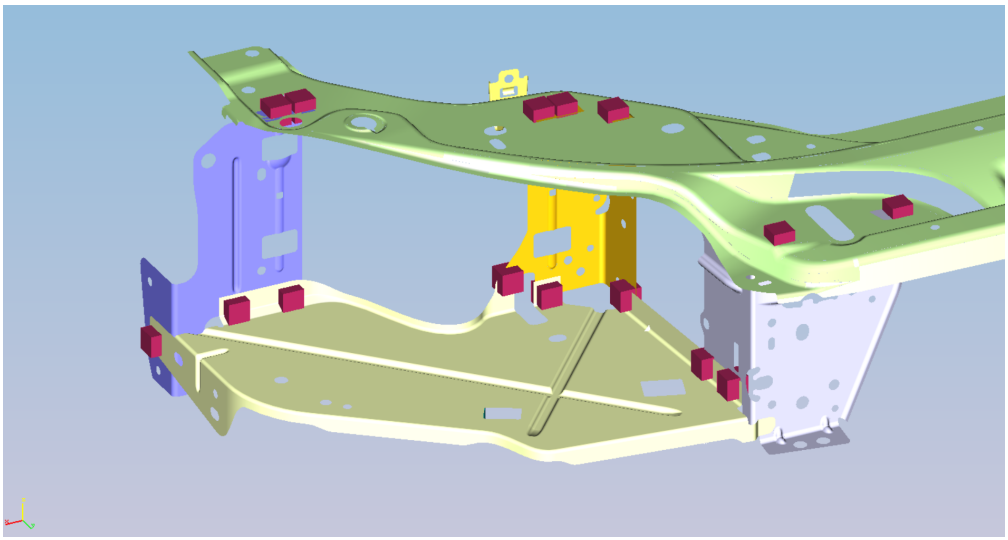
ponenterna som användes i sekvensen som kördes. Det var exempelvis intressant att veta huruvida greppverktyget och frontdetaljer kolliderade innan gripverktyget greppade detaljen. Det var även av vikt att veta om roboten någon gång slog i sig själv eller något i sin omgivning. Figur 3.9 visar ett exempel på resultatet från en kollisionssimulering. Röd färg indikerar kollision, medan gul indikerar att två delar är väldigt nära varandra.



Figur 3.9: Bild på ett kollisionstest

All kinematik i simuleringen delades upp i två kategorier, svets- och förflyttningsoperationer. Operationerna byggdes upp av flertalet punkter i simuleringens koordinatsystem som roboten sedan följde. Hur förflyttningen skedde mellan punkterna räknades ut av programvaran, men det fanns flertalet parametrar som styrde hur. Parametrarna ställdes in för att ta hänsyn till hur snabbt roboten kunde färdas och enligt vilken beräkningsmetod under respektive del av programmet.

Svetsning: Vid skapandet av svetsoperationer sattes först de punkter där svetsning skulle ske ut. Justeringar av robotens färdväg krävdes för att den skulle färdas riskfritt och effektivt, därför placerades via-punkter ut längst färdbanan. Svetspunkternas ordning skapades för att en så enkel färdväg som möjligt skulle användas. Denna färdväg planerades med hänsyn till svetspunkternas orientering, både i koordinatsystemet såväl som deras orientering till varandra. Figur 3.10 visar frontdelen i PS med utmarkerade svetspunkter representerade som lila kuber.

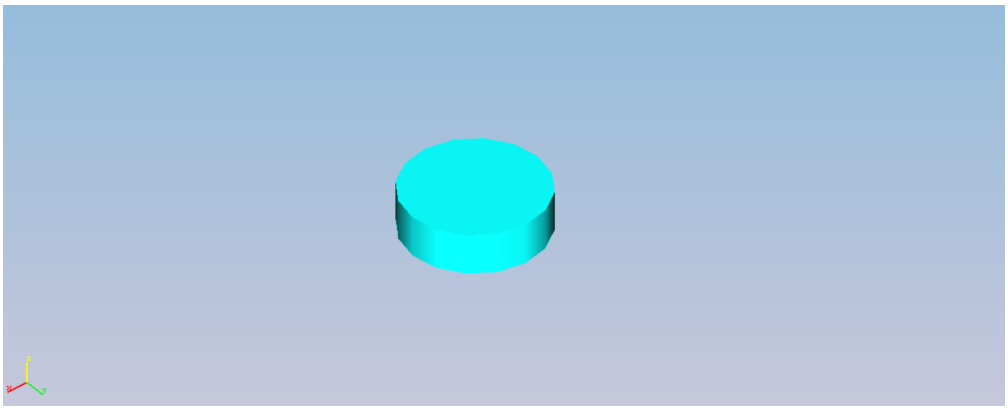


Figur 3.10: Svetspunkter på frontdelarna

Förflyttning: När en förflyttningsoperation skapades sattes upphämtningspunkten och avlämningspunkten ut först. Därefter skapas via-punkter för att säkerställa att hämtning och lämning gjordes på ett säkert och effektivt sätt. I och med skapandet av upphämtningspunkten bestämdes var skapandet av detaljerna i simuleringen uppstår.

3.5 Logik

För att logik skulle kunna byggas upp krävdes det att flertalet sensorer med tillhörande signaler skapades. Kommunikationssignaler för robotarna skapades. Enkla grafiska representationer skapades av sensorerna och placerades ut i den virtuella modellen. Figur 3.11 visar expempel på en grafisk representation av en sensor.

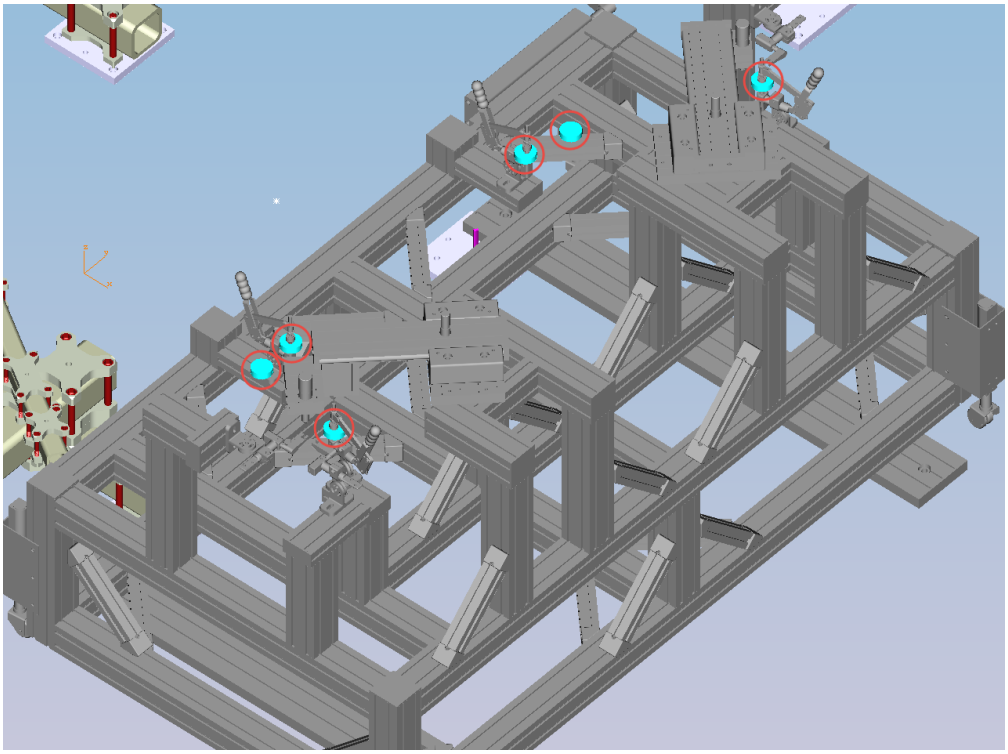


Figur 3.11: Exempel på en grafisk representation av en sensor

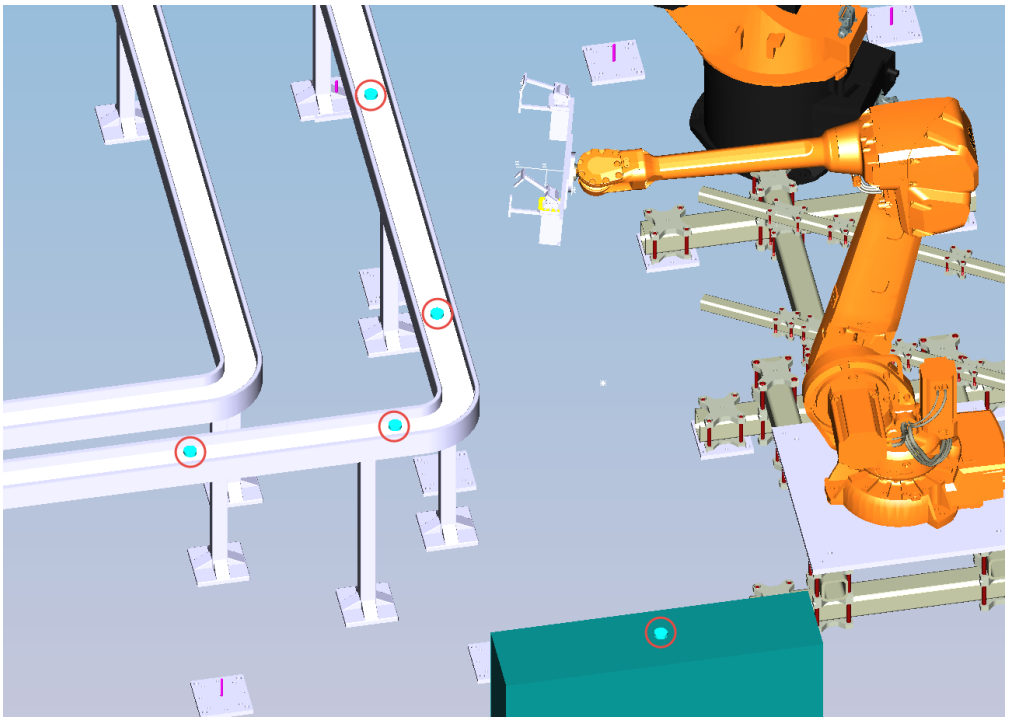
Sensorerna var hårdkopplade till att enbart reagera på en specifik detalj, eller grupp av detaljer, exempelvis känna av om en specifik plåtbit lagts i fixturen. Sensorn krävde en grafisk representation och har en sfärisk interaktion med övriga komponenter.

Dessa sensorer användes för att kontrollera att detaljerna lades i position i fixturen och att de hade kommit till rätt position på transportbandet. Det fanns flera sätt att sedan koppla dessa sensorers signaler, tillsammans med robotarnas signaler, för att skapa logiken bakom simuleringen.

Sensorernas signaler användes för att ge information kring vilket skede produktionen i cellen befann sig i. Dessa användes även för att skapa övergångsvillkor, beskrivna som transistions i PS. Sensorernas placering i fixturen visas i figur 3.12 och sensorerna längs med transportbanan visas i figur 3.13. Sensorerna är markerade med röda cirklar.

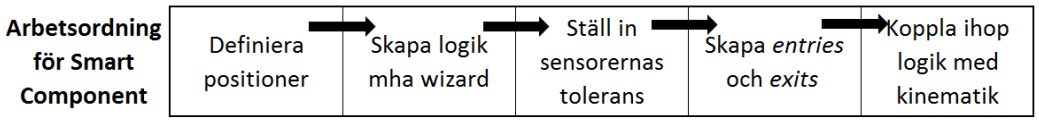


Figur 3.12: Sensorernas placering i fixturen.



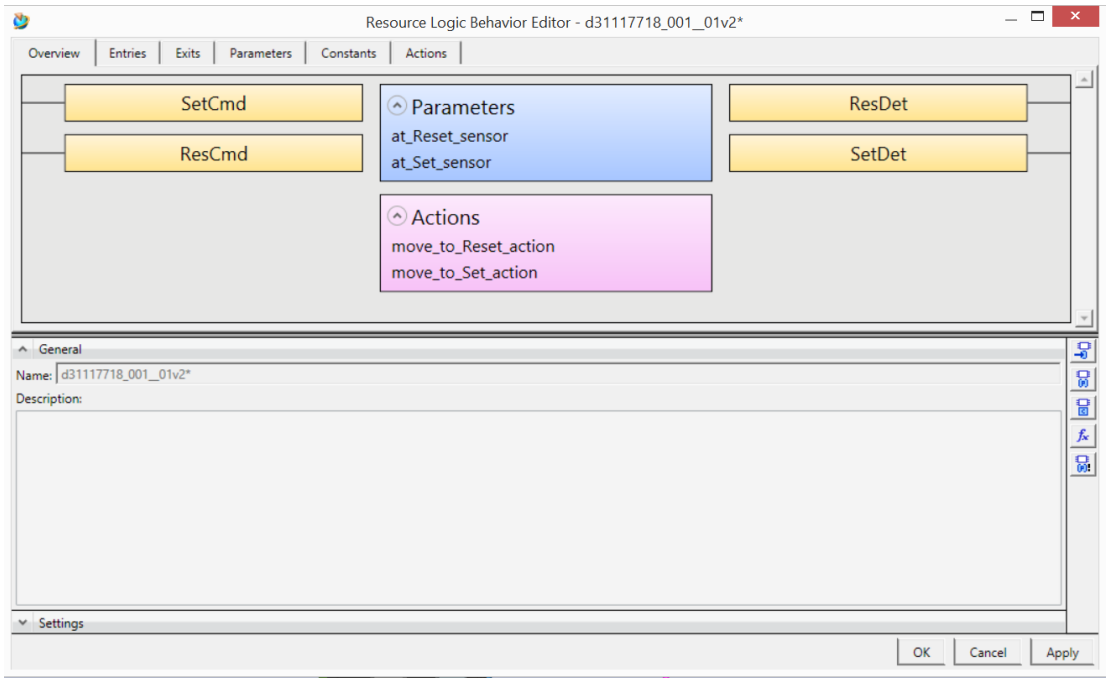
Figur 3.13: Sensorernas placering längs transportbanan och på uppläggningsbordet.

Ett andra verktyg för att skapa logiken i simuleringen var utnyttjandet av smart component. För att skapa logiken och samtidigt kinematiken hos fixturklamparna används detta. Arbetsordningen vid skapande av smart component visas i figur 3.14.



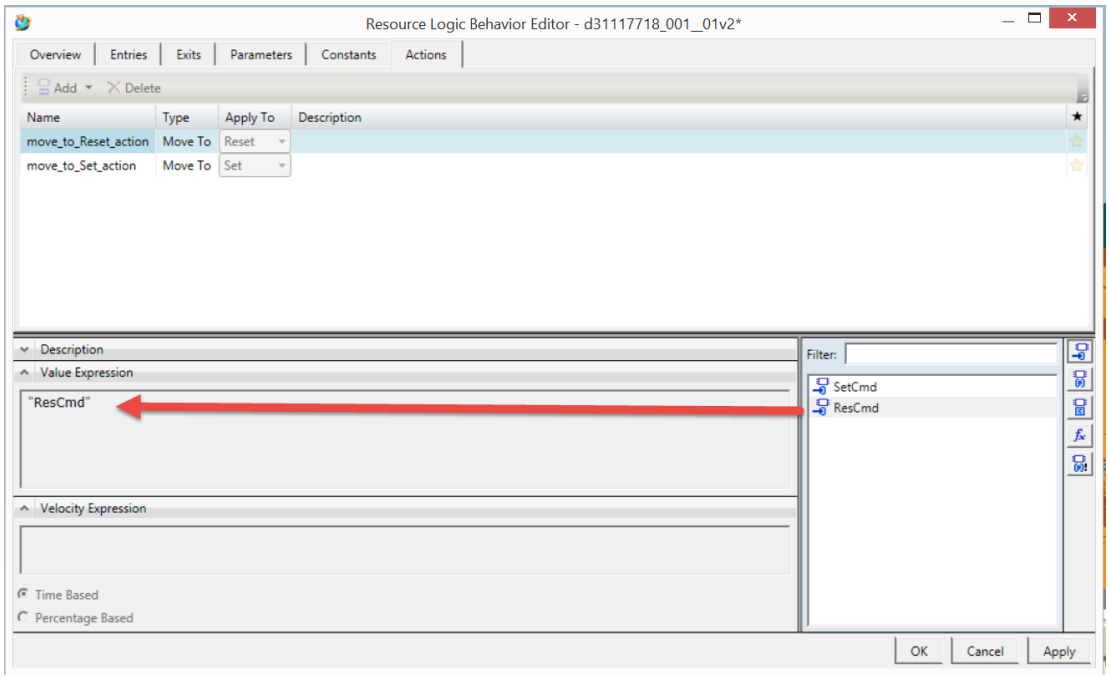
Figur 3.14: Arbetsgång vid skapande av smart component

I wizard som används i steg två skapades rörelserna för att öppna och stänga klampen tillsammans med de signaler som krävs. Om modellen av klampen är korrekt till att börja med kommer ingen vidare ändring behövas av logiken. Figur 3.15 visar ett logikblock skapat av denna wizard , där de gula rutorna på sidorna beskriver in- respektive utsignaler. Den blåa rutan *Parameters* visar sensorerna som modellen har och *Actions* visar de rörelsemönster som modellen innehåller.



Figur 3.15: Logikblock skapat av wizard

Om det skulle krävas finns möjligheten att manuellt knyta samman signalerna med öppna/stänga-kinematiken. Metoden är beskriven i figur 3.16. Signalerna kopplas genom att markera en *Action* och kopplar den till en signal genom att dra in den i fältet *Value Expression*.

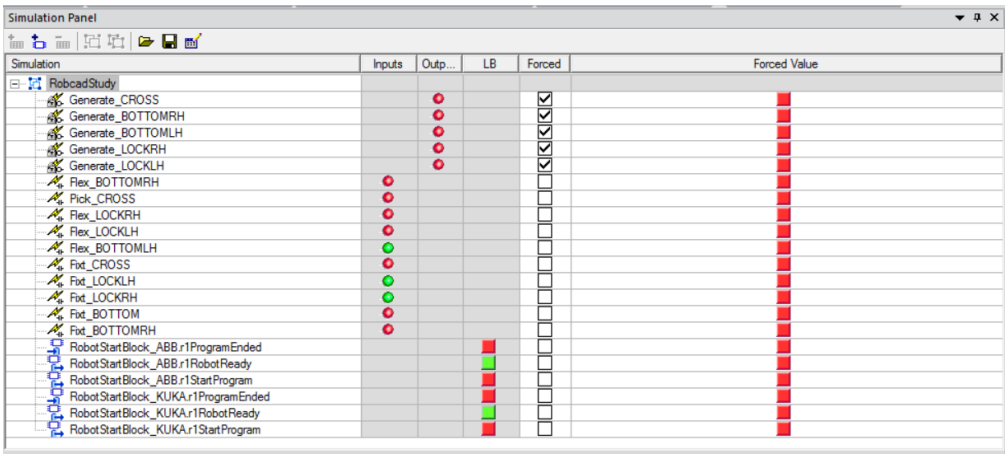


Figur 3.16: Manuell sammankoppling av signaler och kinematik

Robotarna hade flertalet signaler knutna still sig, och dessa utnyttjades i flertalet logikblock, exempelvis StartBlock eller AllocateZone. StartBlock styr villkoren kring hur roboten startar, eller väntar på att starta, en operation. Logikblocket AllocateZone är ett alternativ till PS inbyggda funktion för zonallokering, och ligger i programkoden för att minska risken för kollisioner.

3.6 Simulering

För att styra simuleringen skapades flera icke-kinematiska operationer. Dessa operationers funktion var att i simuleringen generera plåtdetaljerna. Genom att tvinga övergångsvillkoret till dessa operationer kunde inflödet av material till cellen styras. I simuleringspanelen, figur 3.17, kunde även signalernas status avläsas. Genom att låta material flöda i olika sekvenser kunde olika scenarier testas i produktionscellen för att kontrollera att beteendet var korrekt.



Figur 3.17: Simuleringspanelen i PS

3.6.1 Överföring till verkligheten

För att dra nytta av den logik som har ställts upp i simuleringen exporteras denna till en XML-fil som sedan kan användas och implementeras i den verkliga processen. Robotarnas rörelsebeteenden kan exporteras likt programkoden för att användas i de fysiska robotarnas controllers.

4

Resultat

Det största delmålet av att utvärdera användandet av virtuell beredningsteknik och en tidig virtuell beredning av logik, vid framtagning av en tillverkningscell, var att genomföra själva beredningen. En sanningsenlig modell av produktionscellen i PSL har skapats i Process Simulate och en lyckad simulering av nämnda cell har gjorts. Programmeringen är händelsestyrd och fungerar utan att låsa sig. I cellen produceras autonomt fronter till Volvo S80, en i taget. In i cellen kommer tio separata delar och sammanfogas genom punktsvetsning av 28 svetspunkter till en sammansatt bit. Denna bit transporteras ut ur cellen genom att en robot lyfter ut den ur cellen.

Målet att bygga upp en modell av produktionscellen enligt Volvo Cars standarder uppnåddes och utfördes. Modellen av produktionscellen som byggts upp i PD/PS följer Volvo Cars standard för programmering av robotceller vad gäller dess logikuppbyggnad. Samma typ och namn på signaler, signaloperationer och logikblock som Volvo Cars använder används i simuleringen. Objektträden i programmet följer Volvos ramverk för trädstruktur.

Att använda Process Designer och Process Simulate för att skapa PLC-program i tidig beredningsfas är även det gjort enligt uppsatta mål. Det vill säga att koden är verifierad och validerad mot den skapade virtuella modellen av PSL med ett fungerande resultat.

Genom modellering och simulering har det varit möjligt att utvärdera hur virtuell beredning av programlogik och idrifttagning fungerar samt hur det är att använda sig av metoden, vilket var projektet primära mål.

5

Diskussion & slutsatser

Slutsatserna är att virtuell modellering och simulering av produktionsceller kan vara ett mycket kraftfullt hjälpmedel. Det kommer dock inte utan ansträngning och användning av en del resurser.

Fördelar med virtuell beredning av programlogik och virtual commissioning:

- Minskar tiden produktionen behöver stå still vid ändringar.
- Omställningar blir mer flexibla.
- Bättre kvalitet tack vare bättre förberedelser.
- Möjlighet att verifiera och validera programkod innan den fysiska idrifttagningen.
- Kan minska kostnader för större företag.
- Möjligt att prova riskfyllda produktionsmoment.
- Undviker dubbelt arbete med programlogik.

Nackdelar med virtuell beredning av programlogik och virtual commissioning:

- Stora kostnader för inköp av licenser av programvara.
- Stora kostnader för att skaffa sig nödvändig kompetens att kunna arbeta tillfredställande i programvarorna.
- Krävs fortfarande behandling av styrkod efter export ur programmet.

Process Designer och Process Simulate är två avancerade programvaror i sammanhanget. Att bemästra dessa två skulle ge stora möjligheter för företag, men programvarorna är svåra och komplexa. Det krävs mycket träning för att få dem att fungera och bete sig som man önskar. Det är mycket som är möjligt att göra och det är inte alltid helt lätt att förstå hur man utför vissa operationer. Exempelvis måste objekt och komponenter specificeras mycket grundligt för att simulering ska vara möjlig.

En av de stora fördelarna med VC i ett tidigt skede av beredningsfasen är att styrlogik kan verifieras och valideras tidigt. Styrlogiken kan sedan användas vid idrifttagning av produktionscell i verkligheten. Samtidigt kan felsökning av processen göras virtuellt istället för efter uppbyggnad av produktionscellen, vilket minskar både beredningstid och kostnader.

Istället för att ta fram en PLC-kod under beredning av produktionssystemet och en till idifttagningen är det möjligt att virtuellt bereda en PLC-kod som kan användas även vid idifttagningen. Den stora fördelen med att virtuellt bereda användbar logik i en tidig beredningsfas är det sparar tid då framtagning av PLC-kod två gånger undviks.

I det ideala fallet ska inte någon felsökning av koden behövas då den överförs till processens PLC. Verkligheten ser däremot annorlunda ut. Med största sannolikhet kommer det behövas göra en kontroll då det är svårt att göra simuleringen helt sanningsenlig pga. den mänskliga faktorn. I och med att tekniken ständigt utvecklas kommer behovet av en fysisk kontroll minska. Dock kommer behovet inte kunna elimineras.

En programkod har verifierats och validerats mot den virtuella modellen med hjälp av en fungerande simulering. En osäkerhet finns dock i att koden bara verifieras mot den virtuella modellen och inte någon fysisk produktionscell. Det går inte med säkerhet säga att koden är komplett och skulle fungera vid implementering i verkligheten. En bidragande orsak till denna osäkerhet är att inte kompletta verktyg eller fixtur använts. Gripverktyget skapades utan hållfasthetsberäkningar och fixturen som användes är konstruerad för manuell iläggning av detaljer. Sensorernas beteenden är inte samma som en verklig sensor, som kan skilja sig mycket beroende på modell. I simuleringen är sensorerna felfria, vilket kan skilja sig från verkligheten.

Det finns många resonemang som kan föras kring huruvida ett införande av VC i ett företag är ekonomiskt försvarbart. För mindre produktioner är insatsen antagligen för stor för att det ska kunna ge ett tillräckligt utslag på exempelvis time-to-market. På stora företag med stora och komplexa fabriker skulle VC vara en styrka för företaget. Genom att kunna verifiera programkod virtuellt innan den implementeras i en fysisk cell eller att kunna modellera, experimentera och programmera virtuellt skulle en högre kvalitet på programmeringen och designen av produktionssystemet kunna uppnås. Även tidsvinster skulle kunna göras genom kortare uppstartstider. Om man betraktar ett stort företag medför implementering av VC en betydligt större omläggning i metod. Detta inkluderar inte bara fysiska omställningar av exempelvis mjuk- och hårdvara, utan att omställningar ska tas emot och förstås av de anställda som påverkas.

För den produktionscell som projektet har kretsat kring hade antagligen arbetet gått snabbare om en alternativ metod hade använts. Detta för att robotcellen är av låg komplexitet och har ett lågt matrialflöde. Denna slutsats baseras på projektgruppen erfarenheter och uppskattningar kring hur mycket tid framtagningen av styrlogiken och robotprogrammen hade tagit med metoder de redan är familjära med. Den alternativa metoden vi anser skulle vara effektivare är offline-programmering av robot på plats och framtagning av PLC-koden i andra programvaror som använts i tidigare kurser i vår utbildning. Med dessa metoder krävs det dock att produktion stoppas för test och utveckling av programkod vilket ej är önskvärt. Någon uppskattning kring hur stor skillnaden skulle vara är svår att ge.

Det finns flera faktorer som är viktiga att beakta när dessa uppskattningar görs. I detta fall hade vi erfarenhet av en alternativ metod till VC, vilket kan medföra att den altervativa metoden föredras. Om omständigheterna hade varit annorlunda, exempelvis om kunskapen inom VC var i nivå med den befintliga kunskapen inom den alternativa metoden, är det troligt att uppskattningen hade varit annan, till virtual commissionings fördel.

5.1 Rekommendationer för framtida projekt inom samma ämne

För att kunna genomföra en bättre utvärdering av användandet av virtuell beredning av programlogik och VC finns det några faktorer som vi tror skulle medföra ett mer omfattande resultat. Vi känner att projektet har behandlat frågan något ensidigt.

- Projektet bör utföras på samma produktionssystem under samma förutsättningar av två eller flera grupper, där varje grupp arbetar strikt med sin egna specifika beredningsmetod. För att resultatet ska bli så objektivt som möjligt är det viktigt att exempelvis den ingående kompetensen är likställd i varje grupp.

- Flertalet varianter på relevanta produktionssystem bör testas. Produktionssystemen bör variera i storlek och komplexitet. Olika scenarier är även intressant att undersöka, exempelvis om det är ett nytt produktionssystem som ska skapas eller om det är en omstrukturering av befintligt.

- Vi ser att metoden att virtuellt bereda programlogik borde testas och utvärderas genom att arbeta mot verkligheten också. Kod bör överföras för att verifiera att koden även fungerar i en verklig produktionscell. Det skulle även ge erfarenheter och kunskaper i hur försiktig idrifttagningen av en verklig cell borde ske. Detta med tanke på felprogrammeringar och felaktiga robotprogram som kan vara till skada för utrustningen. Metoder i hur fel rättas till, ifall de skulle uppkomma, är också intressanta att utreda.

5.2 Källkritik

Günther Reinhart är professor vid *iwb*, ett institut i München som fick tekniskt universitetstatus år 1869. Där är han chef för institutionen och har en lång rad verk publicerade. Artikeln som refereras till i rapporten som han skrev med Georg Wünsch är en källa med god tillförlitlighet.

Rainer Drath och hans medskribenter må vara forskare vid ABB i Landenburg, Tyskland och därmed riskerar att vinkla artiklar till ABB's fördel. Då faktan från artikeln användes med detta i åtanke och därför plockad med försiktighet med avseende på åsikter anser projektgruppen att informationen är tillförlitlig och trovärdig.

Artikeln med namnet "An evolutionary approach for the industrial introduction of virtual commissioning" skriven av Wolfgang Kühn. Kühn är doktor inom produktionsplanering och -styrning. Hans verk anses vara av tillförlitlig art.

Gary Vasilash är skaparen och chefredaktören på tidningen Automotive Design & Production där han har jobbat i över 27 år. Med en akademisk bakgrund från Eastern Michigan University känns hans fakta trovärdig.

Masterstesen "Virtual Preparation of Tetra Pak Filling Machine", användes för att visa på exempel där virtuell beredning användes ute på företag. En kandidatrapport som heter "Design och styrning av en verklig och virtuell tillverkningscell"har även använts för att påvisa tidigare arbete. Ur denna hämtades information kring vad som gjorts i PSL tidigare.

Artikeln "Engineering method for adaptive manufacturing systems design", skriven av bland annat Marcello Pellicciari användes för att visa ytterligare exempel på hur virtuell beredning använts ute i industrin.

Totalförsvarets forskningsinstituts rapport "Snabbsnigelmetoden: En iterativ metod för utveckling av simuleringsprogram" är en trovärdig källa. Institutet är känt och har ett gott anseende.

Mike Albrecht är inte knuten till något akademiskt institut, men har ett akademiskt förflutet från Colorado State University där han tog en masterexamen. Hans trovärdighet är kanske inte så hög, men hans formuleringar om vad discrete event simulation är akademiska och välformulerade.

Källor från Siemens om Process Designer och Process Simulate samt JT-formatet är vinklade till Siemens fördel. Dock har projektgruppen varit medvetna om detta och därför enbart behandlat och tagit med information som är funktioner programmen klarar av och information som inte innehåller några som helst åsikter eller vinklingar. Även källor från Dassault Systèmes har granskats och sållats som källorna från Siemens.

Litteraturförteckning

[1] G. Reinhart, G. Wünsch, "Economic application of virtual commissioning to mechatronic production systems," i *Production Engineering*, 1:a uppl., vol. 1, D. Biermann, Red. Garching, Tyskland: Springer, 2007, ss. 371–379.
URL `http://dx.doi.org/10.1007/s11740-007-0066-0`

[2] R. Drath, P. Weber, N. Mauser, "An evolutionary approach for the industrial introduction of virtual commissioning", i: *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Hamburg, Tyskland, 15-18 september, IEEE, 2008, ss. 5–8.

[3] W. Kühn, "Digital Factory: Simulation Enhancing the Product and Production Engineering Process", i: *Proceedings of the 38th Conference on Winter Simulation*, WSC '06, 2006, ss. 1899–1906.

[4] G. S. Vasilash, "NOT "ONLY A MOVIE"", *Automotive Design & Production,* vol. 121, nr. 7, s. 32, 2009, [Online] Tillgänglig: Proquest, http://search.proquest.com/docview/217446196?accountid=10041. [Hämtad: 19 maj., 2014].

[5] A. Hollander, S. Sappei, "Virtual Preparation of Tetra Pak Filling Machine", Chalmers tekniska högskola, Göteborg, Sverige, Report No. EX052/2011, 2011.

[6] K. Adzievski, K. Elowson, M. Karlsson, C. Rydberg, H. Sahlin, B. Talani, "Design och styrning av en verklig och virtuell tillverkningscell", Chalmers tekniska högskola, Göteborg, Sverige, Kandidatrapport SSYX02-10-18, 2010.

[7] M. Pellicciari, A. O. Andrisano, F. Leali, A. Vergnano, "Engineering method for adaptive manufacturing systems design," i *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 4:e uppl., vol. 3, X. Fischer, D. Coutellier, Red., Paris, Frankrike: Springer, 2009, ss. 81–91.
URL `http://dx.doi.org/10.1007/s12008-009-0065-9`

[8] Siemens, "Process Designer: Powerful 3D enviroment for manufacturing process planning", *Siemens PLM*, 2011, [Online]. Tillgänglig:

https://www.plm.automation.siemens.com/se_se/products/tecnomatix/robotics_automation/process_designer.shtml., [Hämtad: 19 maj, 2014].

[9] Siemens, ”Process Simulate: Manufacturing process verification in powerful 3D environment”, *Siemens PLM*, 2011, [Online]. Tillgänglig: https://www.plm.automation.siemens.com/se_se/products/tecnomatix/robotics_automation/process_simulate.shtml., [Hämtad: 19 maj, 2014].

[10] S. Henriksson, ”Simulering”, Nationalencyklopedin, 1987, [Online]. Tillgänglig: http://www.ne.se/lang/simulering/305867., [Hämtad: 19 maj, 2014].

[11] B. Jansson, E. Lindberg, M. Persson, ”Snabbsnigelmetoden: En iterativ metod för utvärdering av simuleringsprogram”, Totalförsvaret forskningsinstitut, Stockholm, Sverige, Metodrapport FOI-R–0728–SE, Feb. 2003.

[12] M. Albrecht, ”Introduction to discrete event simulation”, PE (AZ), 2010, [Online]. Tillgänglig: http://www.albrechts.com/mike/DES/Introduction%20to%20DES.pdf. [Hämtad: 19 maj, 2014].

[13] DassaultSystèmes, ”CATIA: Den digitala produktupplevelsen”, *3ds.com*, Dassault Systèmes 2002-2014, [Online]. Tillgänglig: http://www.3ds.com/se/produkter-och-tjaenster/catia/., [Hämtad: 19 maj, 2014].

[14] DassaultSystèmes, ”DELMIA: Digital tillverkning & produktion.”, *3ds.com*, Dassault Systèmes 2002-2014, [Online]. Tillgänglig: http://www.3ds.com/se/produkter-och-tjaenster/delmia/., [Hämtad: 19 maj, 2014].

[15] J. Phelan, ”Siemens’ JT Data Format Accepted as the World’s First ISO International Standard for Viewing and Sharing Lightweight 3D Product Information”, *Siemens PLM*, 17 dec., 2012, [Online]. Tillgänglig: http://www.plm.automation.siemens.com/en_us/about_us/newsroom/press/press_release.cfm?Component=205727&ComponentTemplate=822. [Hämtad: 19 maj, 2014].

[16] Siemens, ”Explore JT Open and discoer your solution”, *Siemens PLM*, 2014, [Online]. Tillgänglig: http://www.plm.automation.siemens.com/en_us/products/open/jtopen/., [Hämtad: 19 maj, 2014].

[17] M. Jivefors, ”Volvo Cars Specification: Manufacturing Simulation and OLP-preparation of robots”, Volvo Cars, Issue 2.8, 2012.

A

Appendix A

ABB IRB 140
Nettovikt: 99 kg
Kapacitet: 6 kg
Räckvidd 810 mm

ABB IRB 1600-8/1.45
Nettovikt: 250 kg
Kapacitet: 8 kg
Räckvidd: 1450 mm

ABB IRB 4600-40/2.55
Nettovikt: 435 kg
Kapacitet: 40 kg
Räckvidd: 2550 mm
Repeterbarhet: 0.05 - 0.06 mm

KUKA KR 30-3
Nettovikt: 665 kg
Kapacitet: 30 kg
Räckvidd: 2033 mm
Repeterbarhet: 0,06 mm