
Titel:

Systemarkitektur for Sorting Industrial Robot

Versionshistorik

Ver.	Dato	Initialer	Beskrivelse
0.1	02-03-12	RHT	Første udkast lavet fra sidste projekt.
0.2	25-05-12	SLT	Tilføjet definitioner og system oversigt.

Indholdsfortegnelse

Introduktion.....	5
Formål.....	5
Referencer.....	5
Definitioner.....	5
Dokumentstruktur og læsevejledning.....	5
Dokumentets rolle i en iterativ udvikling.....	6
System oversigt.....	7
System kontekst.....	7
System introduktion.....	7
Systemets grænseflader.....	7
Grænseflader til personaktører.....	7
Grænseflader til eksterne system aktører.....	8
Grænseflader til hardware aktører.....	8
Grænseflader til software aktører.....	8
Use case view.....	8
Oversigt.....	8
Logisk view.....	8
Oversigt.....	8
Use case realiseringer.....	8
Use-case X: XXX.....	8
Use-case 4: Manuelt Styre.....	8
Beskrivelse.....	10
Process view.....	10
Oversigt over processer.....	10
Implementering.....	10
Kommunikation og synkronisering.....	10
Procesbeskrivelser.....	10
GUI tråd:.....	10
ScriptRunner tråd:.....	11
Logger tråd:.....	11
Info tråd:.....	11
Deployment view.....	12
Oversigt over systemkonfigurationer.....	12
Node beskrivelser.....	12
SCORBOT-ER4u:.....	12
USB-Controller:.....	12
STK500:.....	12
Workstation:.....	13
Sensor:.....	13
Conveyerbelt:.....	13
Weight:.....	13
Development view.....	13
Oversigt.....	13
Komponentbeskrivelser.....	13
Generelle designbeslutninger.....	13
Arkitekturmål og begrænsninger.....	13
Arkitektur mønstre.....	13
Generelle brugergrænsefladeregler.....	14
Fejlhåndtering.....	14

Implementeringssprog og værktøjer.....	14
Implementeringsbiblioteker.....	14
Størrelse og ydelse.....	14
Kvalitet.....	14
Oversættelse	15
Oversættelse-hardware.....	15
Oversættelse-software.....	15
Oversættelse og linkning.....	15
Installation.....	15
Kørsel.....	15
Kørsels-hardware.....	15
Kørsels-software.....	15
Start og stop.....	15
Informationsdisplay.....	16
Bilag.....	16

Introduktion

Formål

Formålet med dette systemarkitektur-dokument er at dokumentere designet af SIR.

De væsentlige aspekter af designet er specificeret heri, og man kan ved at læse dette dokument opnå et overblik over designet.

Referencer

[1] Kravspecifikation

[2] Doxygen generet kode dokumentation.

Definitioner

GUI = Graphical User Interface – også kendt som den grafiske brugergrænseflade.

WPF = Windows Presentation Foundation.

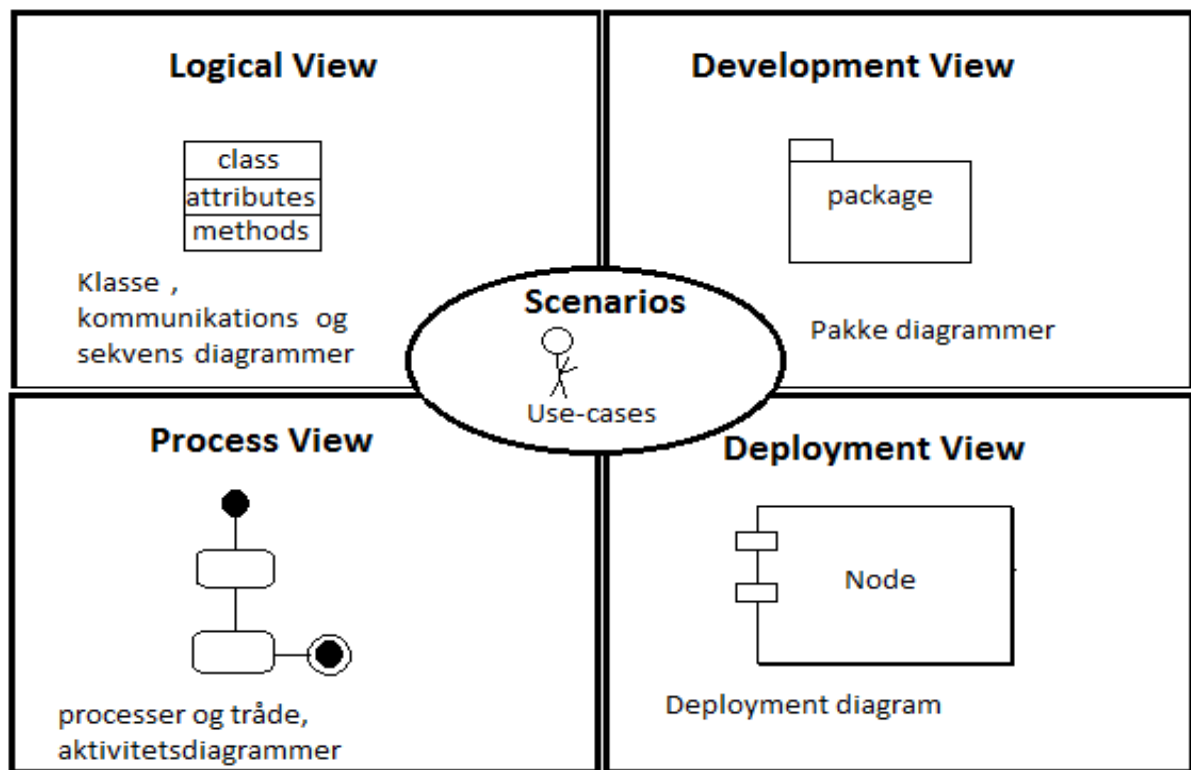
3D = Tre dimensionelt.

CS = Control System.

Dokumentstruktur og læsevejledning

Vi har taget udgangspunkt i 4+1 modellen, som illustrerer forskellige måder at vise softwarearkitekturen på.

Illustration 1: 4+1 modellen



Dokumentets rolle i en iterativ udvikling

Selve dokumentet består af dokumentationen fra de fire iterationer.

I iterationerne er der arbejdet med:

- Udarbejdelse af use-cases i kravspecifikation.
- Design af use-cases.
 1. Sekvensdiagrammer.
 2. Klassebeskrivelser (Doxygen).
 3. Klassediagrammer.
- Implementering af use-cases.
- Unit test og integrationstest.
- Accepttest udarbejdet ud fra kravspecifikationen.
- Udarbejdelse af projektrapport.

System oversigt

System kontekst

Indsæt aktør-kontekst diagrammet.

System introduktion

Systemet har det formål, at den skal lade en bruger af systemet styre en robot gennem den grafiske brugergrænseflade. Der kan foretages bevægelser og målinger af robotten, som også sker gennem den GUI'en.

Robottens bevægelser kan foretages i et 3D-plan, og her kan robottens hånd åbne og lukke, når den skal tage et objekt. Derudover eksisterer transportbåndet, som også er styrbart fra systemet, og her skal det være muligt at aflæse en indbygget sensor til transportbåndet. Grunden til dette ligger i, at sensoren skal kunne registrere, når der er en klods, der er klar til at blive behandlet af robotten ved vægtmåling vha. vejecellen samt måling af objektets rumfang.

Systemet skal løbende logge systemevents, eftersom det vil være en fordel for brugeren at følge med i, hvad der foretages under systemets kørsel.

Endvidere er det påkrævet af systemet, at det skal være muligt at køre sorteringsprocessen på baggrund af en simulator.

Systemets grænseflader

Grænseflader til personaktører

Grænsefladerne til personaktøren "Bruger" er lavet ud fra Windows' WPF. Det er gjort muligt, at tastaturet og musen kan anvendes til at interagere med den visuelle brugergrænseflade, der er opbygget af et fast CS samt tre faner. Aktøren "Bruger" kræver et login for at anvende CS, og når "Bruger" har logget ind, kan den anvende systemet, dens funktionaliteter og modtage informationer uden begrænsninger.

En evt. gennemgang af fanerne?

Grænseflader til eksterne system aktører

Skal have et kommunikationsdiagrammet mellem STK500 og PC.

Grænseflader til hardware aktører

Tekst.

Grænseflader til software aktører

Tekst.

Use case view

Oversigt

Se use cases under Kravspecifikationen.

Logisk view

Oversigt

Tekst.

Drivere:

Tekst.

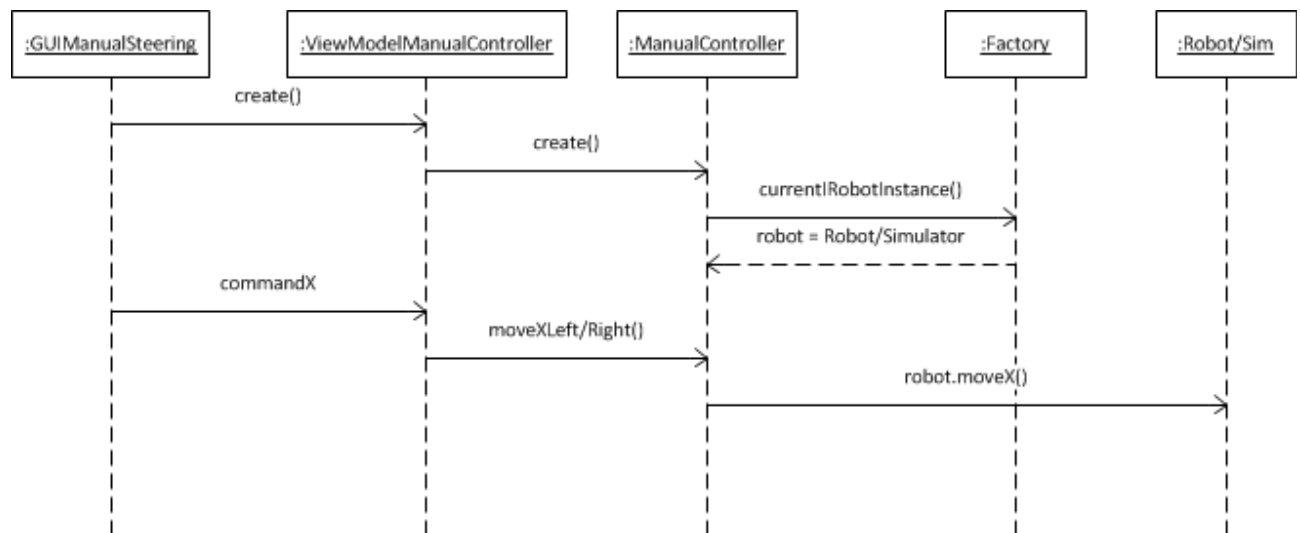
Use case realiseringer

Tekst.

Use-case X: XXX

Noget der forklarer hvad der sker i use casen for eksempel med hjælp af et sekvensdiagram.

Use-case 4: Manuelt Styre



Figur X

Beskrivelse:

Denne er lavet i tre dele:

- View(GUIManualSteering)
- ViewModel(ViewModelManualSteering)
- Model(ManualController med forbindelse videre til IRobot)

I View-delen er der designet, så man har adgang til knapper for de forskellige funktionaliteter for at bevæge robotten. Det er enten ved at dreje en akse til den ene eller anden side eller ved at ændre på en af robotternes koordinater. Den kan også åbne og lukke for kloen.

GUIManualSteering er forbundet med ViewModelManualSteering ved hjælp af normal databinding for hastigheden af bevægelserne, og bevægelsesfunktionerne er blevet implementeret ved hjælp af Commands, der er i ViewModelManualSteering. Der er på denne måde undgået code-behind i View-delen af designet.

ViewModelManualSteering har så simple funktioner som muligt, så der kunne forbindes direkte fra View til ViewModel uden ekstra argumenter. Dette betyder, at der for eksempel er en funktion til at bevæge basen mod højre og en for at bevæge den til venstre.

ManualController har så mere generelle funktioner for bevægelse af robotten, som er forbundet videre til Factory's "currentRobotInstance". Dette betyder, at manuel styring

kan ved kodeeksekvering skifte mellem at blive kørt på Robotten og Simulatoren ude fra styresystemet.

Beskrivelse

Tekst.

Process view

Oversigt over processer

I dette view vises, hvorledes de enkelte processer i systemet kommunikerer. Vi har fire tråde, og herunder ses, hvordan de kommunikerer med hinanden.

[Tegning]

Implementering

Vi har lavet en tråd kontrol klasse(ThreadHandling), hvorfra man laver og kontrollerer trådene i systemet. Dette var for at gøre det nemmere at lave tråde samt lukke dem alle ned, når programmet afsluttes, så der ikke var løse tråde til sidst, der kørte i baggrunden.

Selve trådene er så lavet med standard C# tråde(System.Threading.Thread).

Kommunikation og synkronisering

Til at styre trådene har vi den almindelige join funktion i tråd klassen.

Til synkronisering af loggeren har vi brugt WaitHandle klassen for at kunne signalere, når der er events, der skal logges.

Ved klasserne der bliver delt mellem trådene gennem fabrik klassen(Factory) har vi brugt volatile objekter, og låst (Lock{ }) der hvor man redigerer deres instans referencer. Dette er for at instans referencerne ikke bliver sat flere gange. For eksempel 2 robot forbindelser bliver lavet.

Procesbeskrivelser

GUI tråd:

Beskrivelse:

GUI tråden startes op, når systemet initieres, og det er også vores main tråd, der starter de andre tråde.

Levetid:

Fra program start til slut.

Aktiv:

Aktiv hele tiden.

ScriptRunner tråd:

Beskrivelse:

Bliver kørt hver gang noget kode fra IDE'en skal udføres, så brugeren kan bruge resten af GUI'en, mens koden bliver udført.

Levetid:

Fra koden bliver kørt til den er færdig.

Aktiv:

Mens koden bliver kørt.

Logger tråd:

Beskrivelse:

Bliver startet når loggeren bliver initialiseret og derfra venter den, indtil stoppet, på at den bliver bedt om at logge. Logningen sker ved at der bliver kaldt en funktion på den, som adder en log besked til en liste. Listen bliver der så tjekket i log tråden, hvor de én efter én bliver logget ned.

Levetid:

Fra første gang der logges noget.

Aktiv:

Når der er log beskeder i kø ellers venter den.

[Aktivitets diagram]

Info tråd:

Beskrivelse:

Info tråden bliver brugt hver gang der skal hentes information ned om tabellerne, der vises i GUI'en. Dette er for at resten GUI'en, mens informationen hentes ned eller hvis der ikke er database adgang. Det er teknisk set 2 forskellige tråde, men minder meget om hinanden, den eneste forskel er at den ene henter ned hvilke tabeller der findes, og den anden henter ned selve informationen om tabellen.

Levetid:

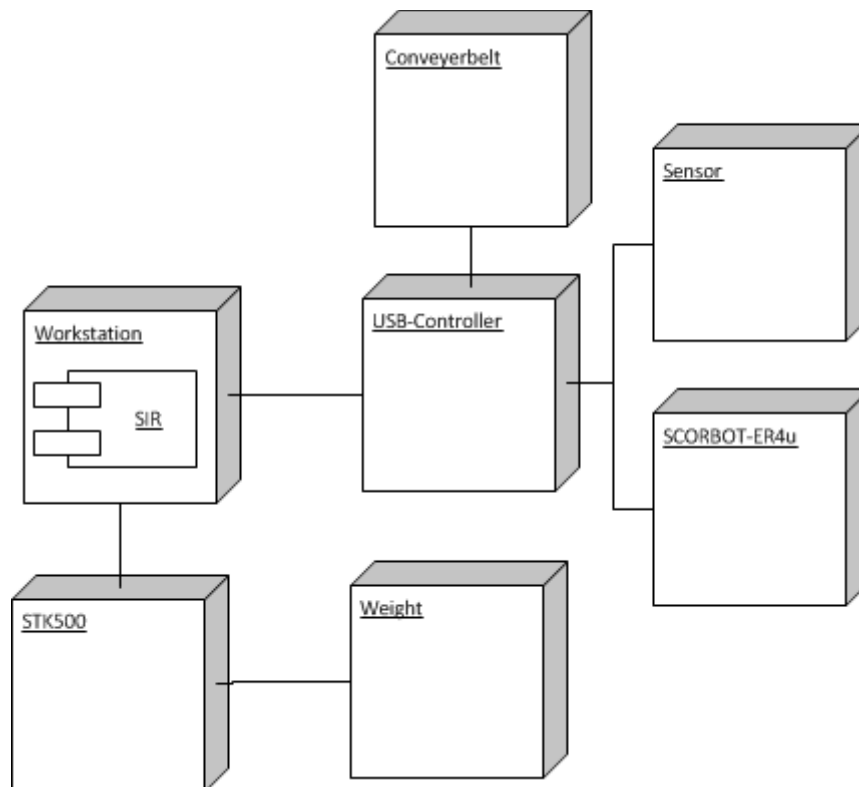
Når brugeren beder om info fra databasen til det er hentet ned.

Aktiv:

Mens data hentes ned.

Deployment view

Oversigt over systemkonfigureringer



Figur beskrivelse

Node beskrivelser

SCORBOT-ER4u:

Robotten som udfører operationerne til sorteringsprocessen.

USB-Controller:

Forbindelse mellem 'workstation' og andre komponenterne som kan styres fra programmet eller hentes værdier fra.

STK500:

Mellemed mellem 'workstation' og 'weight', så der kan tages målinger fra programmet.

Workstation:

Computeren hvor programmet(SIR) køres på.

Sensor:

Lys sensor, som registrerer at der er et objekt eller ej foran den.

Conveyerbelt:

Transportbånd til transport af klodserne, som skal sorteres. På den er monteret 'sensor'.

Weight:

Vægt som der kan tages målinger. Bruges til udregning af massefylde for klodserne.

Development view

Oversigt

Tekst.

Komponentbeskrivelser

Se dokumentation på CD'en for kode dokumentation("kode autogen dokumentation.pdf").

Generelle designbeslutninger

Dette afsnit beskriver de beslutninger vi har taget om arkitekturdesign.

Arkitekturmål og begrænsninger

Som krav og begrænsninger er der blevet sat følgende, der skal overholdes:

- En database skal kunne opbevare og manipulere relevante data fra processen.
- En ID skal kunne fremstille programmer til styring af robotens proces.
- En simulator skal kunne simulere den fysiske robot, således at der kan skiftes mellem den 'ægte' robot og simulatoren.
- Et styreprogram udformet som GUI, der programmeres i C#.
- Et system, der kan måle klodseres masse og rumfang.

Arkitektur mønstre

Vi har valgt at bruge MVVM pattern til at implementere programmet.

Dette er blevet valgt da C# med WPF giver relativ nem mulighed for at implementere med MVVM pattern. Ting som DataBinding og Commands.

Andre patterns der er blevet brugt er Singleton og Factory sammen med Indirection for at fjerne afhængighed fra forskellige komponenter i systemet samt at flere klasser havde brug for at deles om de samme klasse instanser.

Generelle brugergrænsefladeregler

Systemet skal intuitivt, og gennem brugergrænsefladen skal det kunne være muligt for den pågældende bruger at skabe et program, som kan gemmes. Endvidere skal der være mulighed for at styre robotten manuelt gennem GUI'en, og sidst skal der også være mulighed for brugeren at manipulere med de data, der er persisteret i databasen.

Fejlhåndtering

Tekst.

Implementeringssprog og værktøjer

Til databasen:

- Microsoft SQL Server Management Studio
- Database Design Studio Lite (DDS Lite)

Implementeringsbiblioteker

- USBC.dll
- NUnit.
- DotCover.

Størrelse og ydelse

Tekst.

Kvalitet

Tekst.

Oversættelse

Oversættelse-hardware

Tekst.

Oversættelse-software

Selve I4PRJ4 Robot programmet kan afvikles på en Windows maskine med .NET installeret.

Windows maskinen skal være på skolens netværk for at kunne etablere forbindelse til skolens database server, og dermed forbinde til systems database.

For C-programmeringsdelen skal alle kildefiler lægges i samme mappe. I Codevision laves et nyt projekt, som gemmes i samme mappe, og kildefilerne tilføjes projektet. Projektet konfigureres, så det er sat op til Atmega16-chippen og en chipfrekvens på 3,6864 MHz. Derefter kan koden kompileres i Codevision.

Oversættelse og linkning

Tekst.

Installation

Tekst.

Kørsel

Kørsels-hardware

Tekst.

Kørsels-software

Tekst.

Start og stop

Tekst.

Informationsdisplay

Bilag

Tjek medhørende CD for at se bilagene.