

BILAG & KODEBILAG

Tillæg til Hovedopgave

MARG

Modular Applet Robot GUI

Forfattere:

Bjørn Truelsen

Daniel Freiling

Kristina Hansen

Hovedopgave

5. semester, Datamatiker

Roskilde Handelsskole

November 2009

Indhold

1. Projekt Planer	4
1.1 Prerelease	4
1.2 Release 1	5
1.3 Release 2	6
1.4 Release 3	7
2. User-stories	8
2.1 Release 1	8
2.2 Release 2	9
2.3 Release 3	11
3. Task Cards	13
3.1 Release 1	13
3.2 Release 2	15
3.3 Release 3	16
4. CRC Cards	18
ModuleClient	18
AbstractModuleClient	18
XMLClient	18
XMLReader	18
XMLParser	19
ClientData	19
MRCCClient	19
SMRCLParser	19
5. Acceptance Tests	20
5.1 Release 1	20
5.2 Release 2	21
5.3 Release 3	23
6. User-Stories Udvikling	26
6.1 Release 1	26
6.2 Release 2	30
6.3 Release 3	34
7. Kørte Acceptance Tests	41
7.1 Release 2	41
7.2 Release 3	42
8. Sekvens Diagrammer	45
8.1 Release 1	45
8.2 Release 2	51
9. Brugergrænsefladetest	53
9.1 Think aloud test	53
9.2 Spørgeskema	54
10. Referat af DTU Møder	55
10.1 Release 1 d. 24/09/09	55
10.2 Release 2 d. 8/10/09	56
10.3 Release 3 d. 29/10/09	58
11. GUIDE: Creating plugins for MARG	59

11.1 Creating a plugin to parse XML.....	59
11.2 Creating a plugin for a module	61
11.3 Starting from Templates	62
12. Projekt Kontrakt.....	63

1. Projekt Planer

1.1 Prerelease

ID	Task Name	Start	Finish	Duration	sep 2009				
					7	8	9	10	11
1	Pre-Release	07-09-2009	11-09-2009	5d	[Gantt bar from 07-09-2009 to 11-09-2009]				
2	XP teori	07-09-2009	09-09-2009	3d	[Gantt bar from 07-09-2009 to 09-09-2009]				
3	Ekstra Artefakter Teori	09-09-2009	09-09-2009	1d	[Gantt bar on 09-09-2009]				
4	Business Analyse	10-09-2009	10-09-2009	1d	[Gantt bar on 10-09-2009]				
5	Kvalitets faktorer	10-09-2009	10-09-2009	1d	[Gantt bar on 10-09-2009]				
6	Risici til XP Projektet	10-09-2009	10-09-2009	1d	[Gantt bar on 10-09-2009]				
7	Domain Model	11-09-2009	11-09-2009	1d	[Gantt bar on 11-09-2009]				
8	Evaluering af Pre-Release	11-09-2009	11-09-2009	1d	[Gantt bar on 11-09-2009]				

1.2 Release 1

ID	Task Name	Start	Finish	Duration	sep 2009											
					14	15	16	17	18	19	20	21	22	23	24	25
1	Release 1	14-09-2009	25-09-2009	10d												
2	Projekt Plan	14-09-2009	14-09-2009	1d												
3	Metafor	14-09-2009	14-09-2009	1d												
4	User stories	14-09-2009	14-09-2009	1d												
5	Spike investigations	14-09-2009	15-09-2009	2d												
6	Release plan	15-09-2009	15-09-2009	1d												
7	Iteration Plan	15-09-2009	15-09-2009	1d												
8	Task Cards	16-09-2009	16-09-2009	1d												
9	CRC Cards	16-09-2009	16-09-2009	1d												
10	Design Klasse Diagram	16-09-2009	16-09-2009	1d												
11	Acceptance test	16-09-2009	16-09-2009	1d												
12	Sekvens Diagrammer	16-09-2009	16-09-2009	1d												
13	DTU besøg	17-09-2009	17-09-2009	1d												
14	Programming	14-09-2009	25-09-2009	10d												
15	Refactoring	22-09-2009	23-09-2009	1d												
16	Testing	22-09-2009	23-09-2009	1d												
17	Release på DTU	24-09-2009	24-09-2009	1d												
18	Velocity Chart	25-09-2009	25-09-2009	1d												
19	Burn Down Chart	25-09-2009	25-09-2009	1d												
20	Evaluering af Release 1	25-09-2009	25-09-2009	1d												

1.3 Release 2

ID	Task Name	Start	Finish	Duration	sep 2009			okt 2009										
					28	29	30	1	2	3	4	5	6	7	8	9		
1	Release 2	28-09-2009	09-10-2009	10d														
2	Evt ændringer/mangler til dokumentet	28-09-2009	30-09-2009	3d														
3	Indledning	28-09-2009	28-09-2009	1d														
4	Projekt Plan	28-09-2009	28-09-2009	1d														
5	User Stories	28-09-2009	28-09-2009	1d														
6	Revideret Release Plan	28-09-2009	28-09-2009	1d														
7	Iterationplan	29-09-2009	30-09-2009	1d														
8	Task Cards	29-09-2009	30-09-2009	1d														
9	Acceptance Tests	29-09-2009	30-09-2009	1d														
10	Selvens Diagrammer	29-09-2009	30-09-2009	1d														
11	Programmering	30-09-2009	07-10-2009	5d 4h														
12	Opdatere Task Cards	05-10-2009	05-10-2009	1d														
13	Testing	05-10-2009	05-10-2009	1d														
14	Refaktoring	05-10-2009	05-10-2009	1d														
15	Release på DTU	08-10-2009	08-10-2009	1d														
16	Køre Acceptance Tests	08-10-2009	08-10-2009	1d														
17	Evaluering af Release på DTU	09-10-2009	09-10-2009	1d														
18	Velocity Chart	09-10-2009	09-10-2009	1d														
19	Burn Down Chart	09-10-2009	09-10-2009	1d														
20	Evaluering af Release 2	09-10-2009	09-10-2009	1d														

1.4 Release 3

ID	Task Name	Start	Finish	Duration	okt 2009											
					19	20	21	22	23	24	25	26	27	28	29	30
1	Release 3	19-10-2009	30-10-2009	10d												
2	Evt ændringer/mangler til dokumentet	19-10-2009	20-10-2009	2d												
3	Indledning	19-10-2009	19-10-2009	1d												
4	Projekt Plan	19-10-2009	19-10-2009	1d												
5	Revideret Release Plan	19-10-2009	19-10-2009	1d												
6	Iterationplan	19-10-2009	19-10-2009	1d												
7	Task Cards	20-10-2009	20-10-2009	1d												
8	Acceptance Tests	20-10-2009	20-10-2009	1d												
9	Sekvens Diagrammer	29-09-2009	30-09-2009	1d												
10	Programmering	20-10-2009	27-10-2009	6d												
11	Opdatere Task Cards	26-10-2009	27-10-2009	2d												
12	Testing	27-10-2009	29-10-2009	3d												
13	Refaktoring	27-10-2009	27-10-2009	1d												
14	Release på DTU	29-10-2009	29-10-2009	1d												
15	Køre Acceptance Tests	29-10-2009	29-10-2009	1d												
16	Evaluering af Release på DTU	29-10-2009	29-10-2009	1d												
17	Velocity Chart	30-10-2009	30-10-2009	1d												
18	Burn Down Chart	30-10-2009	30-10-2009	1d												
19	Evaluering af Release 3	30-10-2009	30-10-2009	1d												

2. User-stories

2.1 Release 1

Følgende er de ekstra user-stories i Release 1

Forbind til Modul

Forbind til modul
Story Number: 1 Estimation: 10 pts (baseret på spike 1) Business value: High
Brugeren forbinder til et modul på roboten igennem GUI'en. GUI'en informerer brugeren om forbindelsen blev oprettet og viser den umiddelbart efterfølgende kommunikation.
Overvejelser: Denne user-story er meget omfattende, og indebærer hovedfunktionaliteten for vores system. Ved iterationsplanlægning skal denne deles op i mange individuelle tasks. Det at forbinde til et modul indebærer også at modtage XML dataen løbende og gøre den tilgængelig for GUI'en.

Vis variabel-træ

Vis variabel-træ
Story Number: 2 Estimation: 2 pts Business value: High
Brugeren beder om at få præsenteret alle variabler fra et tilsluttet modul. Variablerne bliver vist som en træstruktur.
Overvejelser: Dette er en grafisk præsentation af den interne datastruktur på det givne modul. En meget vigtig funktionalitet med stor business value for kunden.

Send kommando til Modul

Send kommando til modul
Story Number: 3 Estimation: 1 pts (baseret på spike 2) Business value: Medium
Brugeren skriver en kommando i et tekstfelt og vælger at den skal sendes til modulet. Det skal være muligt at se den sendte kommando samt, det rå modtagne data i et vindue.
Overvejelser: Da det at have en forbindelse åben til modulet tilhører en anden user-story, forventer vi ikke at denne user-story vil kræve særligt meget at implementere.

Tilføj variabel til status

Tilføj variabel til status
Story Number: 4
Estimation: 4 pts
Business value: Medium
Brugeren markerer en variabel, og tilføjer den til status. Status er en række variabler der bliver overvåget og hjælper brugeren til at se status for robotten.
Overvejelser: Sværhedsgraden af implementeringen afhænger af om brugeren skal kunne angive grænseværdier for status variabler og hvordan brugeren vil have dette til at fungere.

Tilføj modul

Tilføj modul
Story Number: 5
Estimation: 6 pts
Business value: Medium
Brugeren vælger at tilføje et modul inde fra GUI'en. Konfigurationsinformation omkring modulet angives af brugeren og tilføjes derefter til GUI'en. Det skal være muligt at vælge specielle modul faneblade, så som "Images" osv.
Overvejelser: Afhængig af konfiguration kan det tilføjede modul automatisk forbinde til robot-modulet. Størstedelen af arbejdet ved at implementere denne user-story, forventer vi vil være at tilføje specielle modul faneblade.

2.2 Release 2

Reviderede User-stories

Tilføj variabel til Status

Tilføj variabel til status
Story Number: 4
Estimation: 2 pts
Business value: Medium
Brugeren markerer en variabel, og tilføjer den til status. Status er en række variabler der bliver overvåget og hjælper brugeren til at se status for robotten.
Overvejelser: Sværhedsgraden af implementeringen afhænger af om brugeren skal kunne angive grænseværdier for status variabler og hvordan brugeren vil have dette til at fungere.

Tilføj Modul

Tilføj modul
Story Number: 5
Estimation: 4 pts
Business value: Medium
Brugeren vælger at tilføje et modul inde fra GUI'en. Konfigurationsinformation omkring modulet angives af brugeren og tilføjes derefter til GUI'en. Det skal være muligt at vælge specielle modul faneblade, så som "Images" osv.
Overvejelser: Afhængig af konfiguration kan det tilføjede modul automatisk forbinde til robot-modulet. Størstedelen af arbejdet ved at implementere denne user-story, forventer vi vil være at tilføje specielle modul faneblade.

Nye User-stories

Vis grafer for SMR variabler

Vis grafer for SMR variabler
Story Number: 6
Estimation: 4
Business value: High
Brugeren får præsenteret SMR robottens status ved hjælp af grafer. En graf for linie-sensor og en for den infrarøde-sensor. Der vises også grafisk om hjulene på robotten kører.
Overvejelser:

Juster SMR variabler

Juster SMR variabler
Story Number: 7
Estimation: 2
Business value: High
Brugeren kan justere værdier for hjulene på robotten, og se at de kører.
Overvejelser:
Der skal være en slider bar for hvert af de 2 baghjul.

Ændre variabelværdi i variabeltræ

Ændre variabelværdi i variabeltræ
Story Number: 8
Estimation: 2
Business value: Medium
Brugeren skal kunne højreklikke på en variabel i variabeltræ fanebladet for et modul og vælge at ændre den markerede variabels værdi.
Overvejelser: Det er måske en god idé også at tillade F2 som genvejstast til at redigere en variabels værdi.

Kontrol knapper

Kontrol knapper
Story Number: 9
Estimation: 3 pts
Business value: High
Brugeren skal på hvilket som helst tidspunkt kunne styre robotten fra GUI. Dette gøres med Start, Stop og Pause knapper.
Overvejelser: Projektpartner foreslog at have en genvejs-tast til pause-knappen, så den på hvilket som helst tidspunkt hurtigt kan aktiveres.

Modul status lampe

Modul status lampe
Story Number: 10
Estimation: 2
Business value: Medium
Brugeren skal på hvilket som helst tidspunkt i GUI'en kunne se en status lampe for hvert modul. Denne lampe skal indikere om modulet er forbundet og ellers fungerer som det skal.
Overvejelser: Lampen skal være rød hvis ingen forbindelse. Gul hvis forbindelse, men status variabler forkerte. Grøn hvis både forbindelse og status er OK.

Genbrug af sendte kommandoer

Genbrug af sendte kommandoer
Story Number: 11
Estimation: 1
Business value: Medium
Brugeren skal kunne genbruge tidligere sendte kommandoer til robotten i Kontrol fanebladet. Dette skal ske vha. en drop-down boks med de tidligere kommandoer
Overvejelser: Der skal ikke være auto-completion på, det skal blot være muligt at vælge et tidligere svar med piletasterne. Måske det er bedst kun at vise de 5 nyeste kommandoer i drop-down boksen.

Titel

Titel
Story Number: 12
Estimation: 1
Business value: Low
Browser-vinduet har robotens navn som titel og der kommer et pop-up vindue ved start af applet der siger velkommen til <Robotens Navn>
Overvejelser: Titel på browser-vindue skal "hard-codes" i html filen der aktiverer systemets applet. Popup vinduet kan være en simpel JOptionPane.

2.3 Release 3

Justér SMR variabler

Juster SMR variabler
Story Number: 7
Estimation: 2 pts
Business value: High
Brugeren kan justere værdier for hjulene på robotten, og se på grafen at de kører.
Overvejelser: Der skal være en slider bar for hvert af de 2 baghjul.

Tilføj plugin til modul

Tilføj plugin til modul
Story Number: 13
Estimation: 1 pts
Business value: High
Brugeren kan tilføje et plugin til et modul
Overvejelser: Der skal være en knap der hedder "Add plugin", som tilføjer et plugin.

Tilføj variabel til status

Tilføj variabel til status
Story Number: 4
Estimation: 2 pts
Business value: Medium
Brugeren markerer en variabel, og tilføjer den til status. Status er en række variabler der bliver overvåget og hjælper brugeren til at se status for robotten.
Overvejelser: Sværhedsgraden af implementeringen afhænger af om brugeren skal kunne angive grænseværdier for status variabler og hvordan brugeren vil have dette til at fungere.

Modul status lampe

Modul status lampe
Story Number: 10
Estimation: 2 pts
Business value: Medium
Brugeren skal på hvilket som helst tidspunkt i GUIen kunne se en status lampe for hvert modul. Denne lampe skal indikere om modulet er forbundet og ellers fungerer som det skal.
Overvejelser: Lampen skal være rød hvis ingen forbindelse. Gul hvis forbindelse, men status variabler forkerte. Grøn hvis både forbindelse og status er OK.

Kontrol knapper

Kontrol knapper
Story Number: 9
Estimation: 3 pts
Business value: High
Brugeren skal på hvilket som helst tidspunkt kunne styre robotten fra GUI. Dette gøres med Start, Stop og Pause knapper.
Overvejelser: Projektpartner foreslog at have en genvejs-tast til pause-knappen, så den på hvilket som helst tidspunkt hurtigt kan aktiveres.

Titel

Titel
Story Number: 12
Estimation: 1 pts
Business value: Low
Browser-vinduet har robottens navn som titel og der kommer et pop-up vindue ved start af applet der siger velkommen til <Robottens Navn>
Overvejelser: Titel på browser-vindue skal "hard-codes" i html filen der aktiverer systemets applet. Pop-up vinduet kan være en simpel JOptionPane.

3. Task Cards

3.1 Release 1

Følgende er vores Task Cards fra Release 1.

Lav JUnit test til XMLClient

Task Card 1, Iteration 1, User-story: <u>Forbind til Modul</u>
Dato: 16-9/09
Task: Lav JUnit test til XMLClient
Tid: 2 time
Kommentar:

Lav XMLClient

Task Card 2, Iteration 1, User-story: <u>Forbind til Modul</u>
Dato: 15-9/09
Task: Lav XMLClient
Tid: 4 timer
Kommentar:

Send kommando til modul

Task Card 3, Iteration 1, User-story: <u>Send kommando til modul</u>
Dato: 15-9/09
Task: Send Kommando til modul
Tid: 1 timer
Kommentar:

Lav prototype af overordnet GUI

Task Card 4, Iteration 1, User-story: <u>Forbind til Modul</u>
Dato: 16-9/09 – 21-9/09
Task: Lav prototype af overordnet GUI
Tid: 8 timer
Kommentar:

Læsning af XML

Task Card 5, Iteration 2, User-story: <u>Forbind til Modul</u>
Dato: 16-9/09
Task: Læsning af XML
Tid: 3 timer
Kommentar:

Gøre læsning af XML modulært

Task Card 6, Iteration 2, User-story: <u>Forbind til Modul</u>
Date: 22-9/09
Task: Gøre læsning af XML modulært
Tid: 3 timer
Kommentar:

Opbevar data modtaget fra XML

Task Card 7, Iteration 2, User-story: <u>Forbind til Modul</u>
Date: 21-9/09 – 22-9/09
Task: Opbevar Data modtaget fra XML
Tid: 2 timer
Kommentar:

Gør opbevaring af Data modulær

Task Card 8, Iteration 2, User-story: <u>Forbind til Modul</u>
Date: 22-09/09
Task: Gøre Opbevaring af Data modulær
Tid: 3 timer
Kommentar:

Lav Variabel-træ

Task Card 9, Iteration 2, User-story: <u>Vis variabel-træ</u>
Dato: 16-9/09
Task: Lav variabel-træ
Tid: 2 time
Kommentar: <ul style="list-style-type: none">- Brugte Javas JTree, lavede en ny klasse VariableTree- VariableTree tager imod et oprettet JTree og "dekorerer" det med den data der modtages fra VarDataPlugin.- Man kan i GUI Builderen sætte et normalt JTree ind og så bare dekorere det med vores VariableTree i koden efterfølgende.

3.2 Release 2

Lav et modul plugin

Task Card 10, Release 2, User-story: <u>Vis grafter for SMR variabler</u>
Date:
Task: Lav et modul plugin til SMR robotten
Tid: 2 t.
Kommentar:

Tegn GUI til SMR plugin

Task Card 11, Release 2, User-story: <u>Vis grafer for SMR variabler</u>
Date:
Task: Tegn GUI til SMR plugin
Tid: 3 t.
Kommentar:

Gør det muligt at justere SMR variabler så det kan ses om hjulene kører

Task Card 12, Release 2, User-story: <u>Justér SMR variabler</u>
Date:
Task: Gør det muligt at justere SMR variabler, så det kan ses om hjulene kører.
Tid: 2 t.
Kommentar:

Gør det muligt fra GUI at tilføje et modul

Task Card 13, Release 2, User-story: <u>Tilføj modul</u>
Date:
Task: Gør det muligt fra GUI at tilføje et modul
Tid: 1 t.
Kommentar:

Tilføj XML konfiguration af moduler

Task Card 14, Release 2, User-story: <u>Tilføj modul</u>
Date:
Task: Tilføj XML konfiguration af moduler
Tid: 2 t.
Kommentar:

Tilføj plugin til modul

Task Card 15, Release 2, User-story: <u>Tilføj modul</u>
Date:
Task: Tilføj plugin til modul.
Tid: 3 t.
Kommentar:

Gør det muligt at have forbindelse til flere moduler

Task Card 16 , Release 2, User-story: <u>Tilføj modul</u>
Date:
Task: Gør det muligt at have forbindelse til flere moduler
Tid: 3 t.
Kommentar:

Gør det muligt at ændre variabelværdi i variabeltræ

Task Card 17, Release 2, User-story: <u>Ændre variabelværdi i variabeltræ</u>
Date: 2-10-09
Task: Gør det muligt at ændre variabelværdi i variabeltræ
Tid: 2 t.
Kommentar:

Gør det muligt at genbruge sendte kommandoer

Task Card 18, Release 2, User-story: <u>Genbrug af sendte kommandoer</u>
Date:
Task: Gør det muligt at genbruge sendte kommandoer.
Tid: 1 t.
Kommentar:

3.3 Release 3

Gør det muligt at justere SMR variabler så det kan ses om hjulene kører

Task Card 12, Release 3, User-story: <u>Justér SMR variabler</u>
Date:
Task: Gør det muligt at justere SMR variabler, så det kan ses om hjulene kører.
Tid: 3 t.
Kommentar:

Tilføj plugin til modul.

Task Card 15 , Release 3, User-story: <u>Tilføj plugin til modul</u>
Date:
Task: Tilføj plugin til modul.
Tid: 2 t.
Kommentar:

Lav JUnit til at tjekke grænseværdier for status variabler

Task Card 19 , Release 3, User-story: <u>Tilføj variabel til status</u>
Date:
Task: Lav JUnit til at tjekke grænseværdier for status variabler
Tid: 2 t.
Kommentar:

Lav grafisk variable status side

Task Card 20 , Release 3, User-story: <u>Tilføj variabel til status</u>
Date:
Task: Lav grafisk variabel status side – inkl. mulighed for tilføjelse i variabeltræ
Tid: 1,5 t.
Kommentar:

Lav JUnit til præsentering af overvågede variabler

Task Card 21 , Release 3, User-story: <u>Tilføj variabel til status</u>
Date:
Task: Lav JUnit til præsenteringen af overvågede variabler.
Tid: 2 t.
Kommentar:

Lav behandling og tjek af status variabler

Task Card 22 , Release 3, User-story: <u>Tilføj variabel til status</u>
Date:
Task: Lav behandling og tjek af status variabler
Tid: 2 t.
Kommentar:

Tilføj mulighed for at tjekke status på et modul

Task Card 23 , Release 3, User-story: <u>Modul status lampe</u>
Date:
Task: Tilføj mulighed for at tjekke status på et modul
Tid: 2 t.
Kommentar:

Lav det konfigurerbart i XML, hvor og hvilke kommandoer der skal sendes.

Task Card 24, Release 3, User-story: <u>Kontrol knapper</u>
Date:
Task: Lav det konfigurerbart i XML, hvor og hvilke kommandoer der skal sendes.
Tid: 2 t.
Kommentar:

Lav et pop-up vindue der fortæller ved start, hvilken robot man har forbindelse til.

Task Card 25 , Release 3, User-story: <u>Titel</u>
Date:
Task: Lav pop-up vindue der fortæller ved start, hvilken robot man har forbindelse til.
Tid: 1 t.
Kommentar:

Lav robottens titel konfigurerbar i XML

Task Card 26 , Release 3, User-story: Titel
Date:
Task: Lav robottens titel konfigurerbar i XML.
Tid: 2 t.
Kommentar:

4. CRC Cards

Følgende er vores CRC cards

ModuleClient

Class Name: ModuleClient	
Superclasses:	
Subclasses: Implemented by AbstractModuleClient	
Responsibilities:	Collaboration:
Fastlægge en kontrakt for alle Modul Klienter	

AbstractModuleClient

Class Name: AbstractModuleClient	
Superclasses: (IF)ModuleClient	
Subclasses: XMLClient, MRCCClient	
Responsibilities:	Collaboration:
Oprette en socket forbindelse til et modul	
Lukke en socket forbindelse til et forbundet modul	
Sende tekst/kommandoer til modulet	

XMLClient

Class Name: XMLClient	
Superclasses: AbstractModuleClient, (IF) ContentHandler	
Subclasses:	
Responsibilities:	Collaboration:
Læse XML løbende og omsætte det til data	XMLReader, KlientData
Sende XML-pakket kommando til modul	

XMLReader

Class Name: XMLReader	
Superclasses:	
Subclasses:	
Responsibilities:	Collaboration:
Læse og konvertere XML	XMLParser

XMLParser

Class Name: XMLParser	
Superclasses:	
Subclasses:	
Responsibilities:	Collaboration:
Modtage udpakket data fra XML Reader	XMLReader

ClientData

Class Name: ClientData	
Superclasses:	
Subclasses:	
Responsibilities:	Collaboration:
Opbevare data mens programmet kører	

MRCClient

Class Name: MRCClient	
Superclasses: Abstract Klient	
Subclasses:	
Responsibilities:	Collaboration:
Læse SMRCL	SMRCL Parser,
Skrive kommando i SMRCL	ClientData

SMRCLParser

Class Name: SMRCLParser	
Superclasses:	
Subclasses:	
Responsibilities:	Collaboration:
Læse og konvertere SMRCL	

5. Acceptance Tests

5.1 Release 1

Følgende er vores Acceptance test fra Release 1 med forklaringer under hver.

Forbind til modul 1: Opret forbindelse til modul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Klient får besked på at skulle forbinde til et modul	Besked fra Server om at forbindelsen er oprettet.	

Forklaring

Denne test skal teste om der kan oprettes forbindelse til et modul. Hvis der er forbindelse til modulet, modtages der en enkel besked med modulets navn fra robotten. Sådan en besked kunne f.eks. se sådan ud: `<auClient name="AuClientNox" version="2.04"/>`. Svaret kommer an på hvilket modul man forbinder til, men kravet er at den modtagne besked indeholder modulets navn og version.

Forbind til modul 2: Konverter XML

Test Trin	Input/Handler	Forventet Output	Resultat
1	Kommando sendes til et forbundet modul Klienten sender: "var core.version"	XML der er konverteret til Strings GUI RawData vinduet viser: <code><var name="core.version" value="x.xx"></var></code>	

Forklaring

Denne test går ud på at teste om det XML kode der kommer tilbage fra modulet, når en kommando er afsendt, er blevet konverteret korrekt via vores XMLParser klasse.

Forbind til modul 3: Gem data

Test Trin	Input/Handler	Forventet Output	Resultat
1	Konverteret XML gemmes i intern hukommelse Klient sender: "var allcopy"	Konverteret XML er gemt i intern hukommelse Alle variabler bliver vist i RawData vinduet.	

Forklaring

Denne test går ud på at teste om det konverteret XML data er blevet gemt korrekt. Dette skal kunne ses i RawData vinduet.

Vis Variabel-træ

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på variabel-træets faneblad i GUIen kaldet "Variables".	Variables faneblad i GUI viser alle variabler som en træstruktur. Om alle variabler er til stede kan tjekkes ved at se om den indeholder alt data der blev modtaget i RawData vinduet.	

5.2 Release 2

Følgende er vores Acceptance test fra Release 2.

Vis grafer for SMR variabler

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på SMR faneblad under et modul	I GUIen vises grafer for ir sensor og line sensor fra SMR robotten.	

Justér SMR variabler 1: Justér højre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for højre hjul	Robottens højre hjul kører fremad	
2	Brugeren justerer slideren tilbage for højre hjul	Robottens højre hjul kører tilbage	

Justér SMR variabler 2: Justér venstre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for venstre hjul	Robottens venstre hjul kører fremad	
2	Brugeren justerer slideren tilbage for venstre hjul	Robottens venstre hjul kører tilbage	

Tilføj modul 1: Tilføj modul fra GUI

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på "Add Module"	Et pop-up vindue vises hvor der kan indtastes host, port og modul navn	
2a	Bruger trykker "Cancel" i pop-up vinduet.	Pop-up vindue lukkes.	
2b	Bruger udfylder information i pop-up vinduet og klikker "OK"	Pop-up vindue lukkes og det nye modul tilføjes om en knap i modul menu til venstre. Knappen har navnet på det nye modul.	
3	Bruger klikker på det nye modul i modul menuen.	I Control fanebladet for det valgte modul kan brugeren se den korrekte host og port.	

Tilføj modul 2: Tilføj modul ved konfigurerings

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugen ændrer XML konfigurationen til host = 10.0.2.2 og port = 24928, og navn = RHD. Derefter starter MARG	GUIen indeholder det nye Modul RHD	
2	Bruger klikker på RHD i modul menuen.	I Control fanebladet for det valgte modul kan brugeren se den korrekte host og port.	

Tilføj modul 3: Tilføj plugin til modul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugen trykker på det lille plus tegn udfor det modul der skal tilføjes et faneblad til.	Et pop-up vindue vises hvor der kan i faneblad navn	

Ændre variabelværdi i variabeltræ

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger højreklikker på en variabel i variabeltræet	En pop-up menu vises hvor der kan vælges "Change Value".	
2	Bruger vælger "Change Value"	Nyt vindue åbnes hvor der kan indskrives ny værdi. Den gamle værdi for variabelen er markeret til nem overskrivning.	
3a	Bruger trykker "Cancel"	Pop-up vindue lukkes.	
3b	Bruger indtaster "2" og trykker OK	Pop-up vindue lukkes og nye værdi står i Variabeltræet.	

Genbrug af sendte kommandoer

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger udfylder en kommando; "var help" i Control fanebladet af et modul og trykker send.	Kommandoen blev tilføjet til drop-down under kommandofeltet og kan vælges igen.	
2	Bruger vælger "var help" fra drop-down og trykker send	Kommando bliver sendt til robotten og bliver ikke tilføjet til drop-down igen.	

5.3 Release 3

Justér SMR variabler 1: Justér højre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for højre hjul	Robottens højre hjul kører fremad	
2	Brugeren justerer slideren tilbage for højre hjul	Robottens højre hjul kører tilbage	

Justér SMR variabler 2: Justér venstre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for venstre hjul	Robottens venstre hjul kører fremad	
2	Brugeren justerer slideren tilbage for venstre hjul	Robottens venstre hjul kører tilbage	

Tilføj plugin til modul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Add plugin"	Der vises et pop-up vindue de plugins der er til rådighed	
2a	Brugeren vælger et plugin og trykker "OK"	Pop-up vindue lukkes og de valgte plugin tilføjes som faneblad til det aktive modul	
2b	Bruger trykker "Cancel"	Pop-up vindue lukkes.	

Tilføj variabel til status 1: Tilføj variabel

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren højreklikker på en variabel i variabeltræet.	En pop-up menu vises hvor der kan vælges "Add to Status"	
2	Brugeren vælgerne "Add to Status"	Variabel bliver tilføjet status i Status faneblad	

Tilføj variabel til status 2: Sæt grænseværdi for variabel

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren højreklikker på variabel i Status Faneblad.	En pop-up menu vises hvor der kan vælges "Change value limit"	
2	Bruger klikker på "Change value limit"	Et vindue vises hvor der kan indtastes grænseværdier for variabelen	
3a	Bruger indtaster grænseværdier og trykker "OK"	Grænseværdier bliver sat og vinduet lukkes	
3b	Bruger trykker "Cancel"	Vinduet lukkes	

Kontrol knapper 1: Afprøv Start

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Start"	Robotten starter	

Kontrol knapper 2: Afprøv Stop

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Stop"	Robotten stopper	

Kontrol knapper 3: Afprøv Pause

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Pauser"	Robotten stopper midlertidigt. Pause knappen ændres til "Resume"	
2	Bruger trykker på "Resume"	Robotter fortsætter sin kørsel.	

Titel

Pre-requisite: Bruger ændrer "name" elementet i robot.xml filen til "SMR2"

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger starter systemet som applet i en browser	GUIen vises og der står i øverste venstre hjørne "Loading.." indtil at konfigurationen er blevet læst. Når konfiguration er blevet læst ændres teksten i øverste venstre hjørne til "SMR2".	

6. User-Stories Udvikling

På opfordring fra anden projekt gruppe har vi valgt at lave et afsnit i Bilag hvor man kan se hele udviklingen, med alle artefakter for de enkelte user-stories.

6.1 Release 1

Forbind til Modul

User-story

Forbind til modul
Story Number: 1
Estimation: 10 pts (baseret på spike 1)
Business value: High
Brugeren forbinder til et modul på robotten igennem GUI'en. GUI'en informerer brugeren om forbindelsen blev oprettet og viser den umiddelbart efterfølgende kommunikation.
Overvejelser: Denne user-story er meget omfattende, og indebærer hovedfunktionaliteten for vores system. Ved iterationsplanlægning skal denne deles op i mange individuelle tasks. Det at forbinde til et modul indebærer også at modtage XML dataen løbende og gøre den tilgængelig for GUI'en.

Task Cards

Task Card 1, Iteration 1, User-story: <u>Forbind til Modul</u>
Dato: 16-9/09
Task: Lav JUnit test til XMLClient
Tid: 2 time
Kommentar: <ul style="list-style-type: none">- JUnit blev lavet til XMLClient, hvilket er vores primære implementering af ModuleClient og den klasse der sikrer forbindelse til et modul på robotten.- Til testning af forbindelsen brugte vi en såkaldt mock-server, der er en hurtig server implementering der hver gang sender det samme resultat tilbage til den tilsluttede klient.- Denne mock-server brugte vi til at bekræfte at vores modul kunne oprette en socket forbindelse og at input/output virkede som det skulle.- Vi lavede også boundary tests på connect(host, port) metoden, hvor vi testede for om port ligger inden for grænseværdien 1025-65535.- Med vores mock-server kunne vi også teste om serveren/modulet modtog den kommando som blev sendt af sted.

Task Card 2, Iteration 1, User-story: <u>Forbind til Modul</u>
Dato: 15-9/09
Task: Lav XMLClient
Tid: 4 timer
Kommentar: <ul style="list-style-type: none">- AbstractModuleClient er en abstrakt klasse og en generel implementering af interfacet ModuleClient, som foreskriver en række metoder for at forbinde til og kommunikere med et modul. AbstractModuleClient er lavet som en fælles implementering af socket

forbindelsen for alle senere implementerede ModulKlienter, så som MRCCClient. Dvs. at man i selve implementeringen af en XMLClient eller MRCCClient i store træk kan abstrahere fra den underliggende socket forbindelse.
<ul style="list-style-type: none">- XMLClient er den første subklasse til AbstractModuleClient og er klienten som skal kunne læse XML data der kommer ind fra modulet. Dette gør den vha. Javas egen XMLReader.

Task Card 4, Iteration 1, User-story: <u>Forbind til Modul</u>
Dato: 16-9/09 – 21-9/09
Task: Lav prototype af overordnet GUI
Tid: 8 timer
Kommentar: <ul style="list-style-type: none">- Prototypen for vores overordnede GUI blev lavet over flere dage.- Første del af prototypen blev vist frem og afprøvet på DTU, som var tilfredse med det vi havde lavet.- Vi afventer nærmere feedback omkring den endelige GUI prototype

Task Card 5, Iteration 2, User-story: <u>Forbind til Modul</u>
Dato: 16-9/09
Task: Læsning af XML
Tid: 3 timer
Kommentar: <ul style="list-style-type: none">- Brugt Javas egen XMLReader klasse til at tolke XML data- Skal være flere overvejelser omkring denne reader, når det skal laves modulært

Task Card 6, Iteration 2, User-story: <u>Forbind til Modul</u>
Date:
Task: Gøre læsning af XML modulært
Tid: 3 timer
Kommentar: <ul style="list-style-type: none">- For at gøre læsningen af XML modulært og samtidigt understøtte en plugin struktur, lavede vi et XMLParsePlugin interface, der foreskriver en række metoder til læsning af XML. Vores egen XMLParser klasse, som er den der implementerer ContentHandler og dermed får alle kald fra XMLReader når der er ny data, er klassen som man tilføjer plugins til. XMLParser vil så videresende det data den modtager fra XMLReader til alle plugins den har fået tilføjet.- Således kunne vi lave et RawDataPlugin og et VarDataPlugin, der begge kunne få adgang til den data de har brug for.- Hvert plugin skal også implementere en setPropertyChangeSupport metode således at det er muligt at lave nye property events fra hvert plugin samtidigt med at man subscriber til dem fra XMLParser klassen.

Task Card 7, Iteration 2, User-story: <u>Forbind til Modul</u>
Date: 21-9/09 -
Task: Opbevar Data modtaget fra XML
Tid: 2 timer
Kommentar: <ul style="list-style-type: none">- I den foreløbige implementering af plugin strukturen styrer plugin'et selv hvordan den opbevarer det modtagne data. De fleste plugins kan lade være med at opbevare noget og

blot sende det videre i en `firePropertyChange()` metode, da der normalt kun er brug for det nyeste data. Hvis data-retention bliver nødvendigt, må det implementeres individuelt i hvert plugin.

Task Card 8, Iteration 2, User-story: Forbind til Modul

Date:

Task: Gøre Opbevaring af Data modulær

Tid: 3 timer

Kommentar:

- Modularitet af det opbevarede data opnås ved at hvert plugin selv styrer hvilket data den bibeholder. Tilgængelse af denne data sker igennem `propertyChange` events, men et plugin kan også skrive sine egne metoder så udefrakommende klasser kan tilgå data på andre måder.

Acceptance Tests

Forbind til modul 1: Opret forbindelse til modul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Klient får besked på at skulle forbinde til et modul	Besked fra Server om at forbindelsen er oprettet.	GODKENDT

Forbind til modul 2: Konverter XML

Test Trin	Input/Handler	Forventet Output	Resultat
1	Kommando sendes til et forbundet modul Klienten sender: "var core.version"	XML der er konverteret til Strings GUI RawData vinduet viser: <var name="core.version" value="x.xx"></var>	GODKENDT med en lille fejl: "xxx" i stedet for "x.xx"

Forbind til modul 3: Gem data

Test Trin	Input/Handler	Forventet Output	Resultat
1	Konverteret XML gemmes i intern hukommelse Klient sender: "var allcopy"	Konverteret XML er gemt i intern hukommelse Alle variabler bliver vist i RawData vinduet.	GODKENDT

Send kommando til Modul

User-story

Send kommando til modul
Story Number: 3
Estimation: 1 pts (baseret på spike 2)
Business value: Medium
Brugeren skriver en kommando i et tekstfelt og vælger at den skal sendes til modulet. Det skal være muligt at se den sendte kommando samt, det rå modtagne data i et vindue.
Overvejelser: Da det at have en forbindelse åben til modulet tilhører en anden user-story, forventer vi ikke at denne user-story vil kræve særligt meget at implementere.

Task Card

Task Card 3, Iteration 1, User-story: <u>Send kommando til modul</u>
Dato: 15-9/09
Task: Send Kommando til modul
Tid: 1 timer
Kommentar: <ul style="list-style-type: none">- Udviklet sammen med XMLClient- Meget simpelt, implementering tog under en time.

Acceptance Test

Om kommando kan sendes testes i Acceptance tests til "Forbind til Modul"

Vis variabel-træ

User-story

Vis variabel-træ
Story Number: 2
Estimation: 2 pts
Business value: High
Brugeren beder om at få præsenteret alle variabler fra et tilsluttet modul. Variablerne bliver vist som en træstruktur.
Overvejelser: Dette er en grafisk præsentation af den interne datastruktur på det givne modul. En meget vigtig funktionalitet med stor business value for kunden.

Task Card

Task Card 9, Iteration 2, User-story: <u>Vis variabel-træ</u>
Dato: 16-9/09
Task: Lav variabel-træ
Tid: 2 time
Kommentar: <ul style="list-style-type: none">- Brugte Javas JTree, lavede en ny klasse VariableTree- VariableTree tager imod et oprettet JTree og "dekorerer" det med den data der modtages fra VarDataPlugin.- Man kan i GUI Builderen sætte et normalt JTree ind og så bare dekorere det med vores VariableTree i koden efterfølgende.

Acceptance Tests

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på variabel-træets faneblad i GUIen kaldet "Variables".	Variables faneblad i GUI viser alle variabler som en træstruktur. Om alle variabler er til stede kan tjekkes ved at se om den indeholder alt data der blev modtaget i RawData vinduet.	GODKENDT

6.2 Release 2

Vis grafer for SMR variabler

User-story

Vis grafer for SMR variabler

Story Number: 6

Estimation: 4

Business value: High

Brugeren får præsenteret SMR robotens status ved hjælp af grafer. En graf for line sensor og en for ir sensor. Der vises også grafisk om hjulene på roboten kører.

Overvejelser:

Task Cards

Task Card 10, Release 2, User-story: Vis grafter for SMR variabler

Date: 2-10-09

Task: Lav et modul faneblad plugin til SMR roboten

Tid: 2 t.

Kommentar:

- Vi lavede et interface ModulePlugin som SMRPlugin implementerede.
- Alle modul faneblad plugin skal have metoderne start og stop plugin, så de ikke initialiserer data før de skal bruges.
- ModuleTabs har en ArrayList Modplugin, der indeholder mange ModulePlugin.

Task Card 11, Release 2, User-story: Vis grafter for SMR variabler

Date: 2-10-09

Task: Tegn GUI til SMR plugin

Tid: 3 t.

Kommentar:

- Vi har lavet klasserne LineGraph og RobotWheel hvor vi med kode tegnede de nødvendige GUI elementer til en bar-graf og et hjul der kan køre rundt
- De to klasser har høj binding så de kun fokuserer på at tegne det de bliver bedt om. De har ingen forbindelser til andre klasser.

Acceptance Test

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på SMR faneblad under et modul	I GUI'en vises grafer for ir sensor og line sensor fra SMR robotten.	GODKENDT

Tilføj modul

User-story

Tilføj modul
Story Number: 5
Estimation: 6 pts
Business value: Medium
Brugeren vælger at tilføje et modul inde fra GUI'en. Konfigurationsinformation omkring modulet angives af brugeren og tilføjes derefter til GUI'en. Det skal være muligt at vælge specielle modul-faneblade, så som "Images" osv.
Overvejelser: Afhængig af konfiguration kan det tilføjede modul automatisk forbinde til robot-modulet. Størstedelen af arbejdet ved at implementere denne user-story, forventer vi vil være at tilføje specielle modul faneblade.

Task Cards

Task Card 13, Release 2, User-story: <u>Tilføj modul</u>
Date: 5-10-09
Task: Gør det muligt fra GUI at tilføje et modul
Tid: 1 t.
Kommentar: <ul style="list-style-type: none"> - Lavede vores egen dialog-boks hvor navn, host og port på et modul kan indtastes og derefter kan vinduet lukkes ned. - Dialogboksen er en subklasse til Javas egen JDialog. - Ændrede ModuleTabs klassen så den kan instantieres med navn, host og port. - Gjorde det muligt for ModuleTabs automatisk at forbinde til et givent modul ved opstart

Task Card 14, Release 2, User-story: <u>Tilføj modul</u>
Date: 6-10-09
Task: Tilføj XML konfiguration af moduler
Tid: 2 t.
Kommentar: <ul style="list-style-type: none"> - Vi lavede en XMLConfigManager klasse som står for selve arbejdet og en XMLConfigHandler der agerer mellemmand mellem klienter og Manager. - Manager står for at hente selve XML filen, læse og omforme den til en ønsket datastruktur. - Som ønsket datastruktur lavede vi en simpel klasse inde i XMLConfigManager kaldet Module, der indeholder navn, host, port, autoconnect og en liste af plugins som Strings. - Vi valgte at konfiguration skulle hentes fra robot.xml som skal ligge ved siden af applet - Der var problemer med at lokalisere robot.xml når projektet kørtes lokalt og skulle testes, derfor lavede vi et par metoder i XMLConfigHandler der detekterer om en URL referer til en lokal fil, og hvis den gør, tilretter den.

Task Card 16 , Release 2, User-story: Tilføj modul
Date: 5-10-09
Task: Gør det muligt at have forbindelse til flere moduler
Tid: 3 t.
Kommentar: <ul style="list-style-type: none"> - Vi har lavet et Interface RobotModule, som en afkobling mellem MainGUI og dens moduler, så den kan tilgå de enkelte på en generel måde. - Vi brugte Javas JPanel Cardlayout til at have alle moduler på et panel, så der kan skiftes i mellem hvilke der bliver vist. - Til modulemenuen hvor modul knapperne bliver tilføjet brugte vi Javas Grid Bag layout for at knapperne nemt kunne tilføjes løbende og ligges i samme kolonne. - Vi lagde panelet med knapperne for hvert modul inde i et scrollpane, så der er understøttelse for mange moduler

Acceptance Test

Tilføj modul 1: Tilføj modul fra GUI

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på "Add Module"	Et pop-up vindue vises hvor der kan indtastes host, port og modul navn	GODKENDT
2a	Bruger trykker "Cancel" i pop-up vinduet.	Pop-up vindue lukkes.	GODKENDT
2b	Bruger udfylder information i pop-up vinduet og klikker "OK"	Pop-up vindue lukkes og det nye modul tilføjes om en knap i modul menu til venstre. Knappen har navnet på det nye modul.	GODKENDT
3	Bruger klikker på det nye modul i modul menuen.	I Control fanebladet for det valgte modul kan brugeren se den korrekte host og port.	GODKENDT

Tilføj modul 2: Tilføj modul ved konfiguration

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugen ændrer XML konfigurationen til host = 10.0.2.2 og port = 24928, og navn = RHD. Derefter starter MARG	GUIen indeholder det nye Modul RHD	GODKENDT
2	Bruger klikker på RHD i modul menuen.	I Control fanebladet for det valgte modul kan brugeren se den korrekte host og port.	GODKENDT

Ændre variabelværdi i variabeltræ

User-story

Ændre variabelværdi i variabeltræ
Story Number: 8
Estimation: 2
Business value: Medium
Brugeren skal kunne højreklikke på en variabel i variabeltræ fanebladet for et modul og vælge at ændre den markerede variabels værdi.
Overvejelser: Det er måske en god idé også at tillade F2 som genvejstast til at redigere en variabels værdi.

Task Card

Task Card 17, Release 2, User-story: <u>Ændre variabelværdi i variabeltræ</u>
Date: 2-10-09
Task: Gør det muligt at ændre variabelværdi i variabeltræ
Tid: 2 t.
Kommentar: <ul style="list-style-type: none">- Brugte Java egne GUI objekter til at lave pop-up vindue, hvor den nye værdi kan indtastes- Det at ændre værdien på en variable er i sin egen metode, så den kan senere bruges til genvejstaster, som fx F2.

Acceptance Test

Ændre variabelværdi i variabeltræ

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger højreklikker på en variabel i variabeltræet	En pop-up menu vises hvor der kan vælges "Change Value".	GODKENDT
2	Bruger vælger "Change Value"	Nyt vindue åbnes hvor der kan indskrives ny værdi. Den gamle værdi for variabelen er markeret til nem overskrivning.	GODKENDT
3a	Bruger trykker "Cancel"	Pop-up vindue lukkes.	GODKENDT
3b	Bruger indtaster "2" og trykker OK	Pop-up vindue lukkes og nye værdi står i Variabeltræet.	GODKENDT

Genbrug af sendte kommandoer

User-story

Genbrug af sendte kommandoer
Story Number: 11
Estimation: 1
Business value: Medium
Brugeren skal kunne genbruge tidligere sendte kommandoer til robotten i Control fanebladet. Dette skal ske vha. en dropdown box med de tidligere kommandoer
Overvejelser: Der skal ikke være auto-completion på, det skal blot være muligt at vælge et tidligere svar med piletasterne. Måske bedst kun at vise de 5 nyeste kommandoer i drop-down boxen.

Task Card

Task Card 18, Release 2, User-story: <u>Genbrug af sendte kommandoer</u>
Date: 5-10-09
Task: Gør det muligt at genbruge sendte kommandoer.
Tid: 1
Kommentar: - Vi brugte JCombobox til at opbevare tidligere sendte kommandoer.

Acceptance Test

Genbrug af sendte kommandoer

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger udfylder en kommando; "var help" i Control fanebladet af et modul og trykker send.	Kommandoen blev tilføjet til drop-down under kommandofeltet og kan vælges igen.	GODKENDT
2	Bruger vælger "var help" fra drop-down og trykker send	Kommando bliver sendt til robotten og bliver ikke tilføjet til drop-down igen.	GODKENDT

6.3 Release 3

Juster SMR variabler

User-story

Juster SMR variabler
Story Number: 7
Estimation: 2 pts
Business value: High
Brugeren kan justere værdier for hjulene på robotten, og se på grafen at de kører.
Overvejelser: Der skal være en slider bar for hvert af de 2 baghjul.

Task Card

Task Card 12, Release 3, User-story: <u>Justér SMR variabler</u>
Date: 20-10-09
Task: Gør det muligt at justere SMR variabler, så det kan ses om hjulene kører.
Tid: 3 t.
Kommentar: <ul style="list-style-type: none">- Implementeret efter projektpartners beskrivelse af encoder variabler- Vi rettede ikke i RobotWheel klassen- Vi lagde selve funktionaliteten af at dreje hjulene ift. encoder værdierne i WheelControl klassen

Acceptance Tests

1: Justér højre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for højre hjul	Robottens højre hjul kører fremad	GODKENDT
2	Brugeren justerer slideren tilbage for højre hjul	Robottens højre hjul kører tilbage	GODKENDT

2: Justér venstre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for venstre hjul	Robottens venstre hjul kører fremad	GODKENDT
2	Brugeren justerer slideren tilbage for venstre hjul	Robottens venstre hjul kører tilbage	GODKENDT

Tilføj plugin til modul

User-story

Tilføj plugin til modul
Story Number: 13
Estimation: 1 pts
Business value: High
Brugeren kan tilføje et plugin til et modul
Overvejelser: Der skal være en knap der hedder "Add plugin" som tilføjer et faneblad.

Task Card

Task Card 15 , Release 3, User-story: <u>Tilføj plugin til modul</u>
Date: 20-10-09
Task: Tilføj plugin til modul
Tid: 2 t.
Kommentar: <ul style="list-style-type: none"> - Modulariteten i de forrige releases gjorde denne task nemmere at implementere end først antaget - Vi brugte en JOptionPane til at vise de mulige plugins og hentede dem fra PluginManager

Acceptance Test

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Add plugin"	Der vises et pop-up vindue de plugins der er til rådighed	GODKENDT
2a	Brugeren vælger et plugin og trykker "OK"	Pop-up vindue lukkes og de valgte plugin tilføjes som faneblad til det aktive modul	GODKENDT
2b	Bruger trykker "Cancel"	Pop-up vindue lukkes.	GODKENDT

Tilføj variable til status

User-story

Tilføj variabel til status
Story Number: 4
Estimation: 2 pts
Business value: Medium
Brugeren markerer en variabel, og tilføjer den til status. Status er en række variabler der bliver overvåget og hjælper brugeren til at se status for robotten.
Overvejelser: Sværhedsgraden af implementeringen afhænger af om brugeren skal kunne angive grænseværdier for status variabler og hvordan brugeren vil have dette til at fungere.

Task Cards

Task Card 19 , Release 3, User-story: <u>Tilføj variabel til status</u>
Date: 21-10-09
Task: Lav JUnit til at tjekke grænseværdier for status variabler
Tid: 2 t.
Kommentar: <ul style="list-style-type: none"> - Lavede en JUnit der tjekker om ValueLimit klassen er korrekt implementeret. - ValueLimit blev designet så den har en min og en maxvalue og så den har en metode der tjekker om et givent tal ligger inden for denne grænse.

Task Card 20 , Release 3, User-story: <u>Tilføj variabel til status</u>
Date: 21-10-09
Task: Lav grafisk variabel status faneblad – inkl. mulighed for tilføjelse i variabeltræ
Tid: 1,5 t.

Kommentar:

- Vi byggede videre på højre-klik menu i variabeltræ og tilføjede "Add to Status"
- Vi tilføjede et faneblad til ModuleTabs hvor de tilføjede variabler bliver vist
- Lavede en StatusVariable der er en slags presenter for ModuleVariable, som blot omskriver dens toString metode.
- Gjorde det muligt at tilføje en grænseværdi til en StatusVariable i Status fanebladet via. en højreklik-menu. Klassen ValueLimit blev implementeret på baggrund af JUnit (Task 19).

Task Card 21 , Release 3, User-story: Tilføj variabel til status

Date: 22-10-09

Task: Lav JUnit til præsenteringen af overvågede variabler.

Tid: 2 t.

Kommentar:

- Lavede en JUnit test der tjekker om klassen MonitoredVariable er implementeret korrekt.
- MonitoredVariable blev lavet til at skulle bestå af en ModuleVariable og en tekst string der skal præsenterer denne modulvariabel på en ønsket måde.

Task Card 22 , Release 3, User-story: Tilføj variabel til status

Date: 22-10-09

Task: Lav behandling og tjek af status variabler

Tid: 2 t.

Kommentar:

- For det ikke blev forvirrende med et status faneblad og et status vindue, kaldte vi dette status vindue for "Monitored Variables".
- Vinduet blev lavet så det indeholder en liste af MonitoredVariable objekter.
- MonitoredVariable klassen blev implementeret efter den udviklede JUnit i Task 21.
- Hver MonitoredVariable indeholder en ModuleVariable og et stykke tekst der bruges til at præsentere værdien af ModuleVariable på en ønsket måde.

Acceptance Tests

1: Tilføj variabel

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren højreklikker på en variabel i variabeltræet.	En pop-up menu vises hvor der kan vælges "Add to Status"	GODKENDT
2	Brugeren vælgerne "Add to Status"	Variabel bliver tilføjet status i Status faneblad	GODKENDT

2: Sæt grænseværdi for variabel

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren højreklikker på variabel i Status Faneblad.	En pop-up menu vises hvor der kan vælges "Change value limit"	GODKENDT
2	Bruger klikker på "Change value limit"	Et vindue vises hvor der kan indtastes grænseværdier for variablen	GODKENDT

3a	Bruger indtaster grænseværdier og trykker "OK"	Grænseværdier bliver sat og vinduet lukkes	GODKENDT
3b	Bruger trykker "Cancel"	Vinduet lukkes	GODKENDT

Modul status lampe

User-story

Modul status lampe
Story Number: 10
Estimation: 2
Business value: Medium
Brugeren skal på hvilket som helst tidspunkt i GUI'en kunne se en status lampe for hvert modul. Denne lampe skal indikere om modulet er forbundet og ellers fungerer som det skal.
Overvejelser: Lampen skal være rød hvis ingen forbindelse. Gul hvis forbindelse, men status variabler forkerte. Grøn hvis både forbindelse og status er OK.

Task Card

Task Card 23 , Release 3, User-story: Modul status lampe
Date: 22-10-09
Task: Tilføj mulighed for at tjekke status på et modul
Tid: 2 t.
Kommentar: <ul style="list-style-type: none"> - Status for hvert modul bliver vist med et billede af et Rødt, Gult eller Grønt symbol på det givne moduls knap i venstre side af GUI'en. - Status blev lavet til at opdatere to gange i sekundet ved at bruge RobotModule's getModuleStatusColor der returnerer Grøn hvis alt er okay, Gul hvis en status variabel er ude for sin grænseværdi og Rød hvis der ingen forbindelse er.

Acceptance Test

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger starter applet med et eller flere moduler tilføjet	Status for hvert modul vises ud for dets knap i GUI's venstre side med en form for lampe eller et billede	GODKENDT
2	Bruger vælger et modul der er forbundet og hvor status variabler er korrekte	Modulets status lampe lyser grøn	GODKENDT
3	Bruger vælger et modul der er forbundet, men hvor status variabler er forkerte	Modulets status lampe lyser gul	GODKENDT
4	Bruger vælger et modul der ikke er forbundet	Modulets status lampe lyser rød	GODKENDT

Kontrol knapper

User-story

Kontrol knapper
Story Number: 9
Estimation: 3
Business value: High
Brugeren skal på hvilket som helst tidspunkt kunne styre om robotten skal køre fra GUI. Dette gøres med Start, Stop og Pause knapper.
Overvejelser: Projektpartner foreslog at have en genvejs-tast til pause-knappen, så den på hvilket som helst tidspunkt hurtigt kan aktiveres.

Task Card

Task Card 24, Release 3, User-story: <u>Kontrol knapper</u>
Date: 23-10-09
Task: Lav det konfigurerbart i XML, hvor og hvilke kommandoer der skal sendes.
Tid: 2 t.
Kommentar: <ul style="list-style-type: none">- Vi byggede videre på XMLConfigManager så den også kan læse et controlButtons element og dens indeholdende start, pause og stop elementer.- Start, pause og stop indeholder en moduleName attribut og en command attribut, der tilsammen fortæller vores system hvilket modul den skal have fat i og hvilken kommando der skal sendes når den givne knap trykkes.

Acceptance Tests

Kontrol knapper 1: Afprøv Start

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Start"	Robotten starter	GODKENDT

Kontrol knapper 2: Afprøv Stop

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Stop"	Robotten stopper	GODKENDT

Kontrol knapper 3: Afprøv Pause

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Pause"	Robotten stopper midlertidigt.	GODKENDT

Titel

User-story

Titel
Story Number: 12
Estimation: 1
Business value: Low
Browser-vinduet har robottens navn som titel og der kommer et pop-up vindue ved start af applet der siger velkommen til <Robottens Navn>
Overvejelser: Titel på browser-vindue skal "hard-codes" i html filen der aktiverer systemets applet. Pop-up vinduet kan være en simpel JOptionPane.

Task Card

Task Card 26 , Release 3, User-story: Titel
Date: 23-10-09
Task: Lav robottens titel konfigurerbar i XML.
Tid: 2 t.
Kommentar: <ul style="list-style-type: none">- Lavede et felt i venstre side af GUIen hvor robottens navn skal stå.- Byggede videre på XMLConfigManager så et "name" element der indeholder navnet på robotten der forbindes til også kan læses. Dette navn bliver vist i førnævnte felt, når konfigurationen er blevet indlæst.

Acceptance Test

Pre-requisite: Bruger ændrer "name" elementet i robot.xml filen til "SMR2"

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger starter systemet som applet i en browser	GUIen vises og der står i øverste venstre hjørne "Loading.." indtil at konfigurationen er blevet læst. Når konfiguration er blevet læst ændres teksten i øverste venstre hjørne til "SMR2".	GODKENDT

7. Kørte Acceptance Tests

7.1 Release 2

Vis grafer for SMR variabler

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på SMR faneblad under et modul	I GUIen vises grafer for ir sensor og line sensor fra SMR robotten.	GODKENDT

Tilføj modul 1: Tilføj modul fra GUI

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger klikker på "Add Module"	Et pop-up vindue vises hvor der kan indtastes host, port og modul navn	GODKENDT
2a	Bruger trykker "Cancel" i pop-up vinduet.	Pop-up vindue lukkes.	GODKENDT
2b	Bruger udfylder information i pop-up vinduet og klikker "OK"	Pop-up vindue lukkes og det nye modul tilføjes om en knap i modul menu til venstre. Knappen har navnet på det nye modul.	GODKENDT
3	Bruger klikker på det nye modul i modul menuen.	I Control fanebladet for det valgte modul kan brugeren se den korrekte host og port.	GODKENDT

Tilføj modul 2: Tilføj modul ved konfigurerings

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugen ændrer XML konfigurationen til host = 10.0.2.2 og port = 24928, og navn = RHD. Derefter starter MARG	GUIen indeholder det nye Modul RHD	GODKENDT
2	Bruger klikker på RHD i modul menuen.	I Control fanebladet for det valgte modul kan brugeren se den korrekte host og port.	GODKENDT

Ændre variabelværdi i variabeltræ

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger højreklikker på en variabel i variabeltræet	En pop-up menu vises hvor der kan vælges "Change Value".	GODKENDT
2	Bruger vælger "Change Value"	Nyt vindue åbnes hvor der kan indskrives ny værdi. Den gamle værdi for variabelen er markeret til nem overskrivning.	GODKENDT
3a	Bruger trykker "Cancel"	Pop-up vindue lukkes.	GODKENDT

3b	Bruger indtaster "2" og trykker OK	Pop-up vindue lukkes og nye værdi står i Variabeltræet.	GODKENDT
-----------	------------------------------------	---	----------

Genbrug af sendte kommandoer

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger udfylder en kommando; "var help" i Control fanebladet af et modul og trykker send.	Kommandoen blev tilføjet til drop-down under kommandofeltet og kan vælges igen.	GODKENDT
2	Bruger vælger "var help" fra drop-down og trykker send	Kommando bliver sendt til robotten og bliver ikke tilføjet til drop-down igen.	GODKENDT

7.2 Release 3

Justér SMR variable 1: Justér højre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for højre hjul	Robottens højre hjul kører fremad	GODKENDT
2	Brugeren justerer slideren tilbage for højre hjul	Robottens højre hjul kører tilbage	GODKENDT

Justér SMR variable 2: Justér venstre hjul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren justerer slideren fremad for venstre hjul	Robottens venstre hjul kører fremad	GODKENDT
2	Brugeren justerer slideren tilbage for venstre hjul	Robottens venstre hjul kører tilbage	GODKENDT

Tilføj plugin til modul

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Add plugin"	Der vises et pop-up vindue de plugins der er til rådighed	GODKENDT
2a	Brugeren vælger et plugin og trykker "OK"	Pop-up vindue lukkes og de valgte plugin tilføjes som faneblad til det aktive modul	GODKENDT
2b	Bruger trykker "Cancel"	Pop-up vindue lukkes.	GODKENDT

Tilføj variabel til status 1: Tilføj variabel

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren højreklikker på en variabel i variabeltræet.	En pop-up menu vises hvor der kan vælges "Add to Status"	GODKENDT

2	Brugeren vælgerne "Add to Status"	Variabel bliver tilføjet status i Status faneblad	GODKENDT
---	-----------------------------------	---	----------

Tilføj variabel til status 2: Sæt grænseværdi for variabel

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren højreklikker på variabel i Status Faneblad.	En pop-up menu vises hvor der kan vælges "Change value limit"	GODKENDT
2	Bruger klikker på "Change value limit"	Et vindue vises hvor der kan indtastes grænseværdier for variabelen	GODKENDT
3a	Bruger indtaster grænseværdier og trykker "OK"	Grænseværdier bliver sat og vinduet lukkes	GODKENDT
3b	Bruger trykker "Cancel"	Vinduet lukkes	GODKENDT

Modul status lampe

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger starter applet med et eller flere moduler tilføjet	Status for hvert modul vises ud for dets knap i GUI's venstre side med en form for lampe eller et billede	GODKENDT
2	Bruger vælger et modul der er forbundet og hvor status variabler er korrekte	Modulets status lampe lyser grøn	GODKENDT
3	Bruger vælger et modul der er forbundet, men hvor status variabler er forkerte	Modulets status lampe lyser gul	GODKENDT
4	Bruger vælger et modul der ikke er forbundet	Modulets status lampe lyser rød	GODKENDT

Kontrol knapper 1: Afprøv Start

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Start"	Robotten starter	GODKENDT

Kontrol knapper 2: Afprøv Stop

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Stop"	Robotten stopper	GODKENDT

Kontrol knapper 3: Afprøv Pause

Test Trin	Input/Handler	Forventet Output	Resultat
1	Brugeren trykker på "Pause"	Robotten stopper midlertidigt.	GODKENDT

Titel

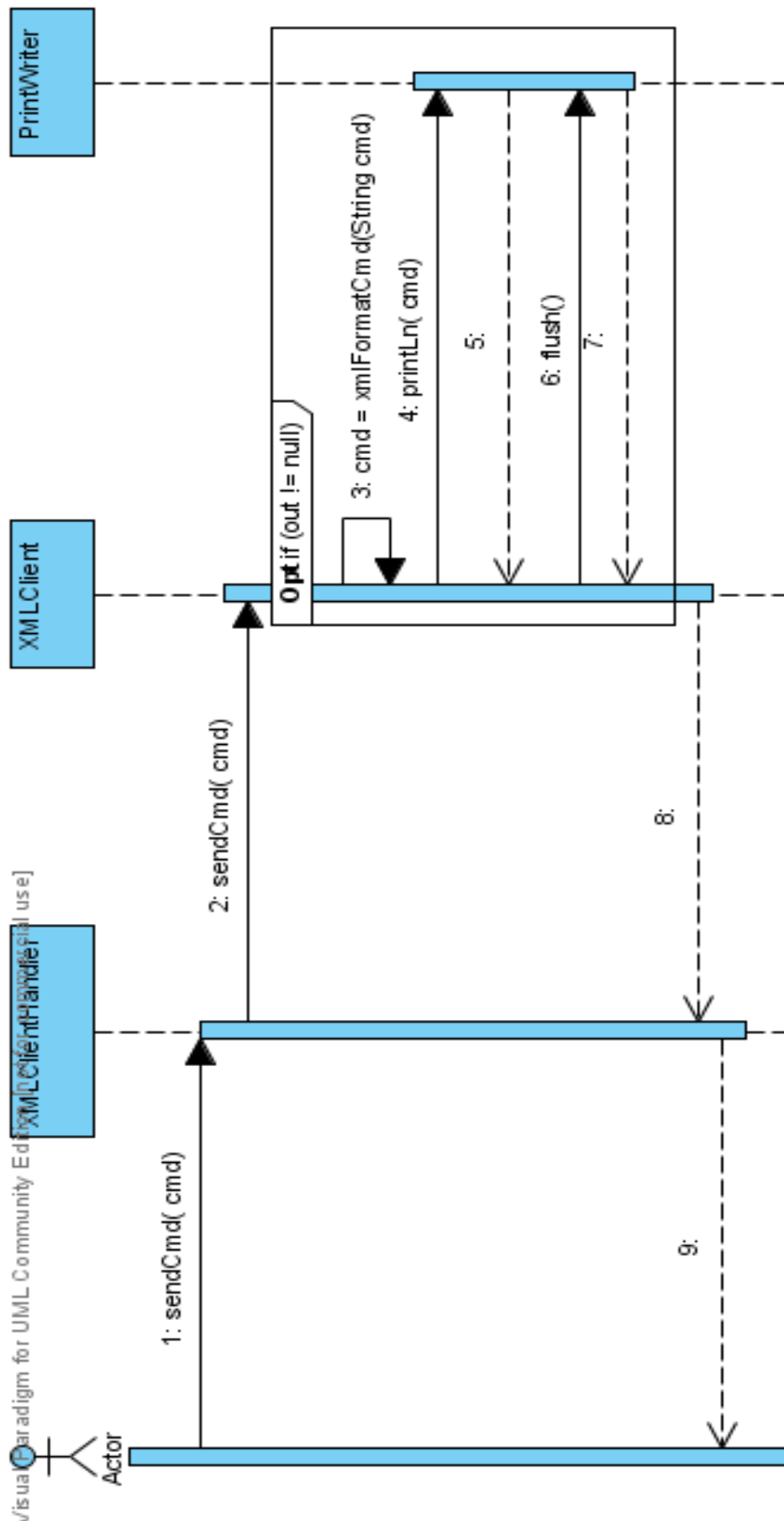
Pre-requisite: Bruger ændrer "name" elementet i robot.xml filen til "SMR2"

Test Trin	Input/Handler	Forventet Output	Resultat
1	Bruger starter systemet som applet i en browser	GUIen vises og der står i øverste venstre hjørne "Loading.." indtil at konfigurationen er blevet læst. Når konfiguration er blevet læst ændres teksten i øverste venstre hjørne til "SMR2".	GODKENDT

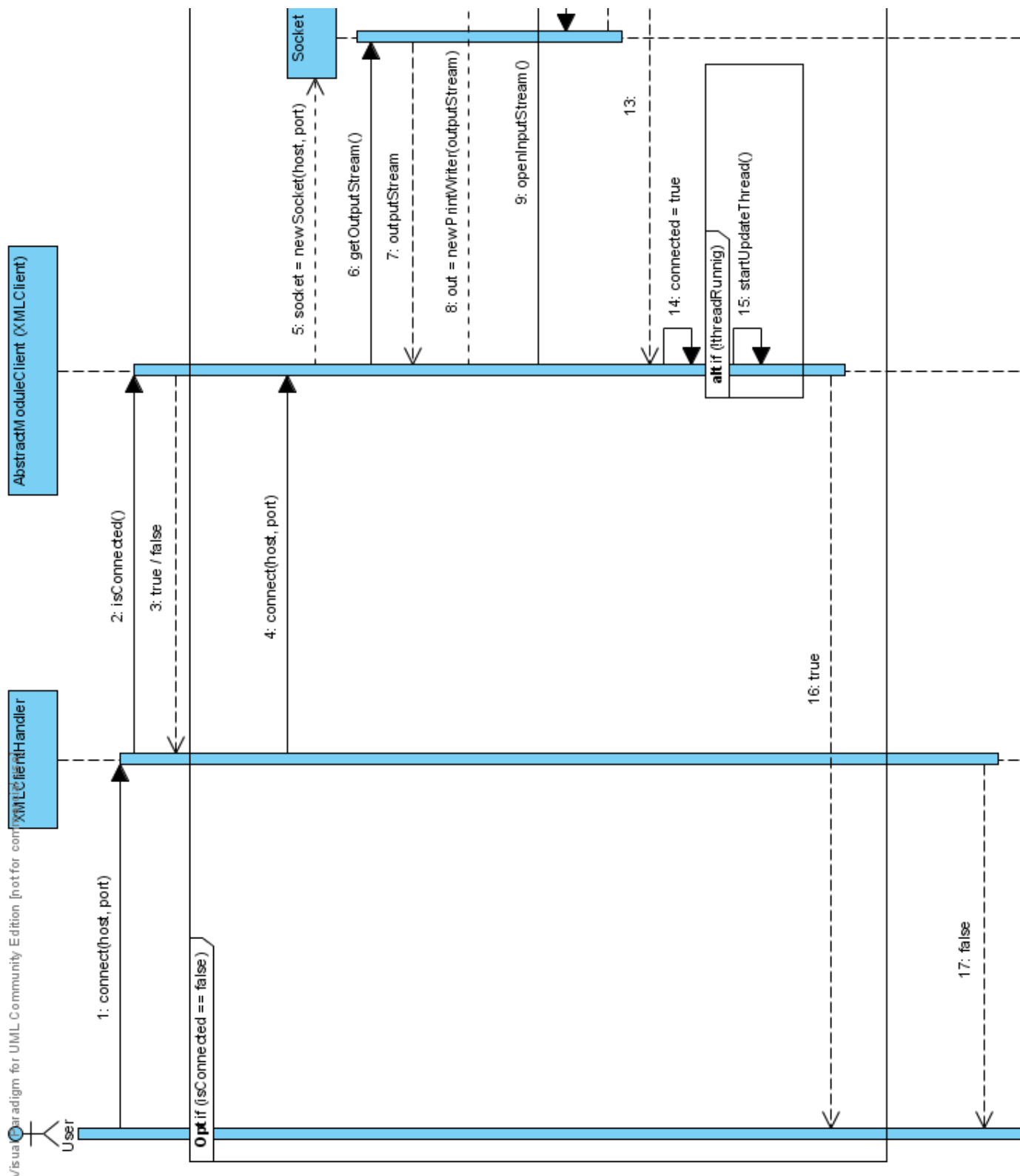
8. Sekvens Diagrammer

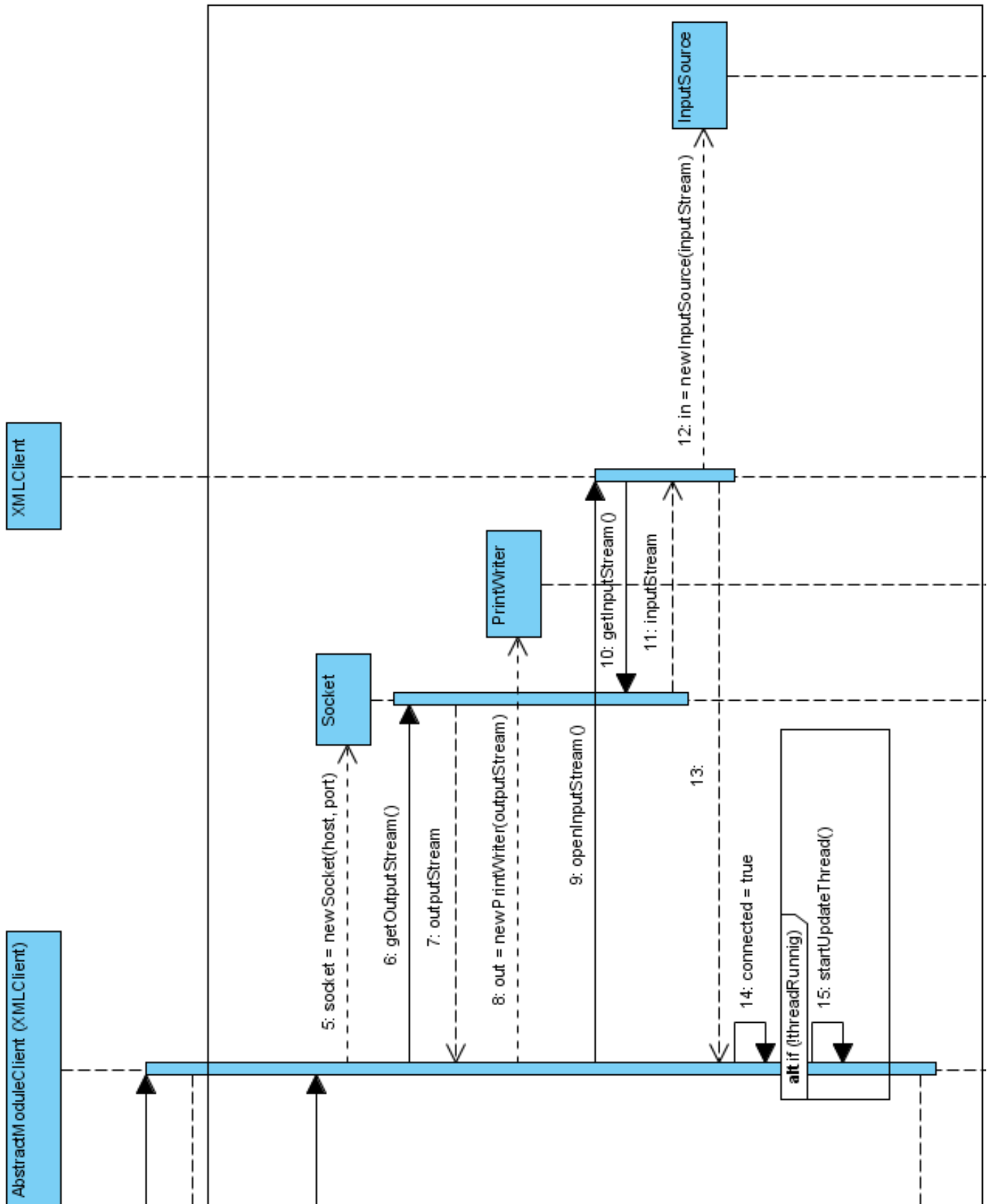
8.1 Release 1

8.1.1 Send kommando til modul

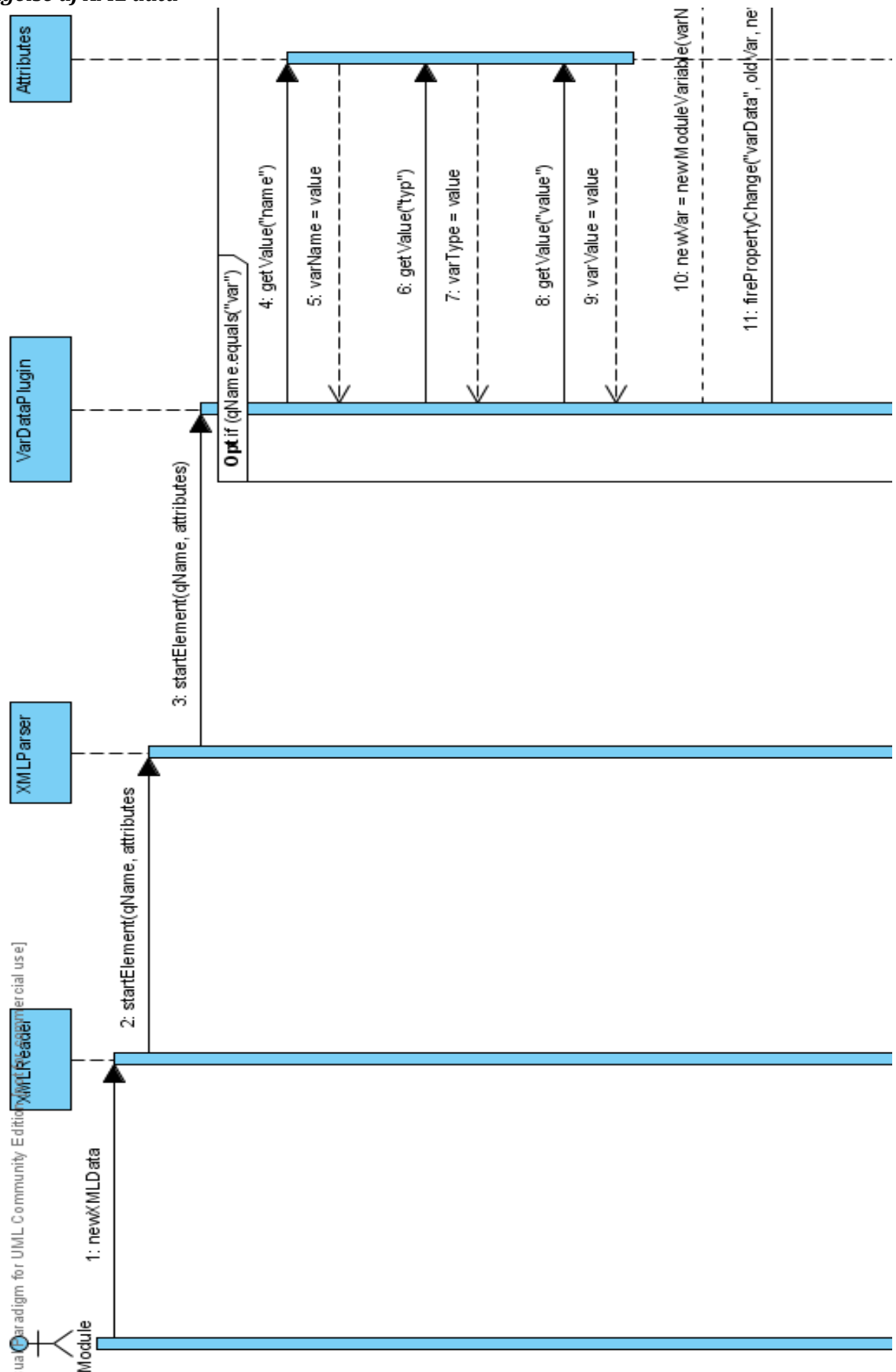


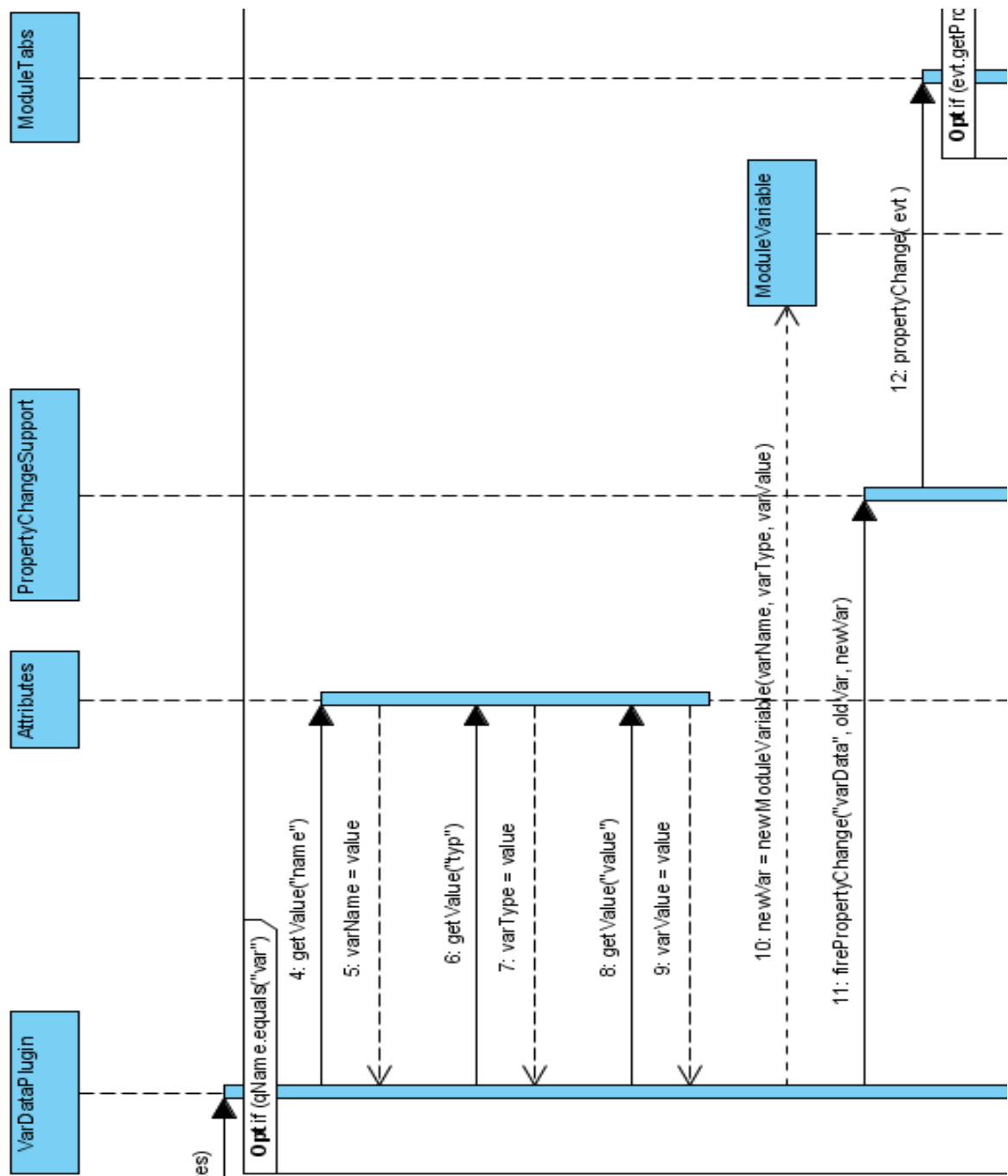
8.1.2 Forbind til Modul

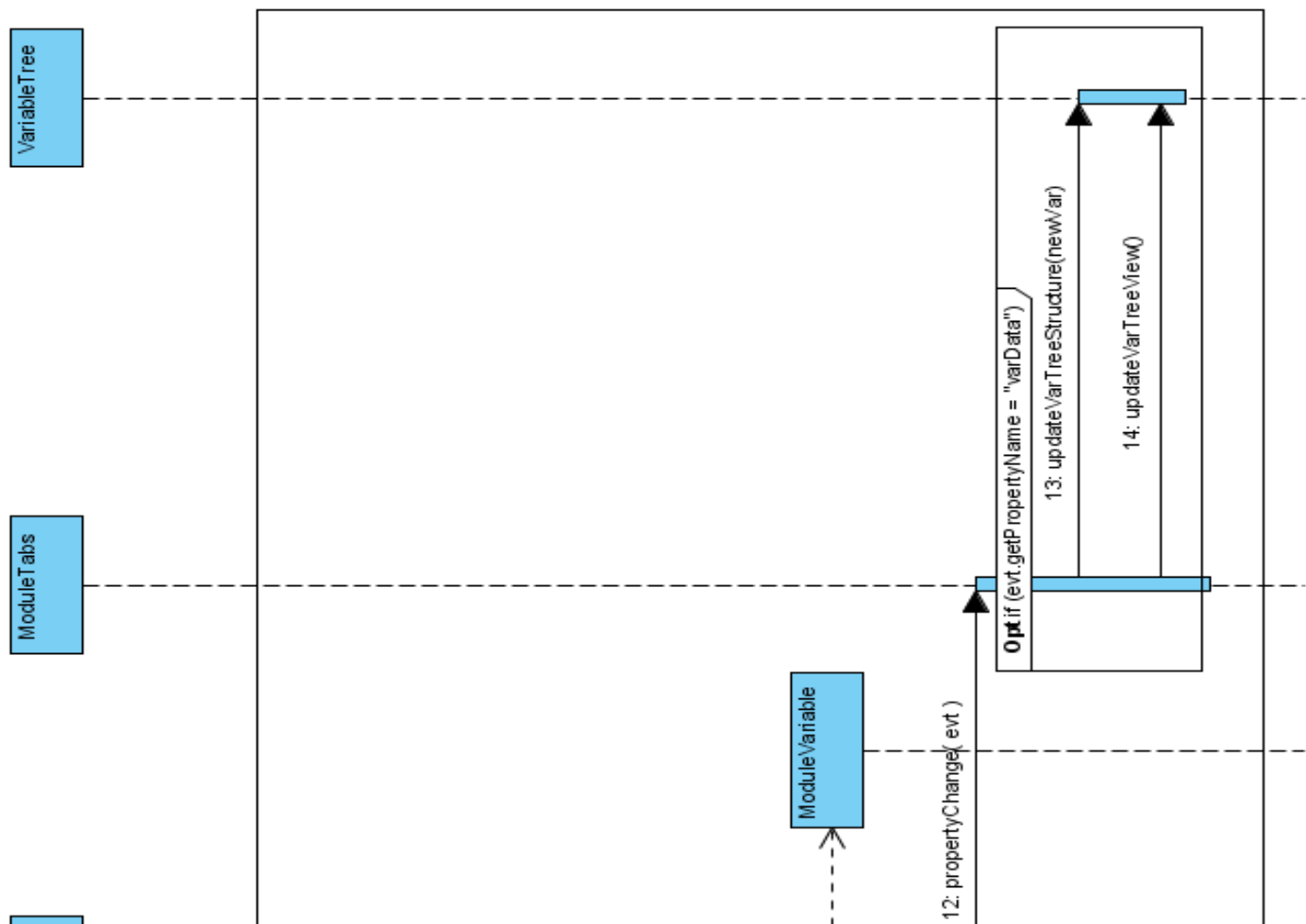




8.1.3 Modtagelse af XML data

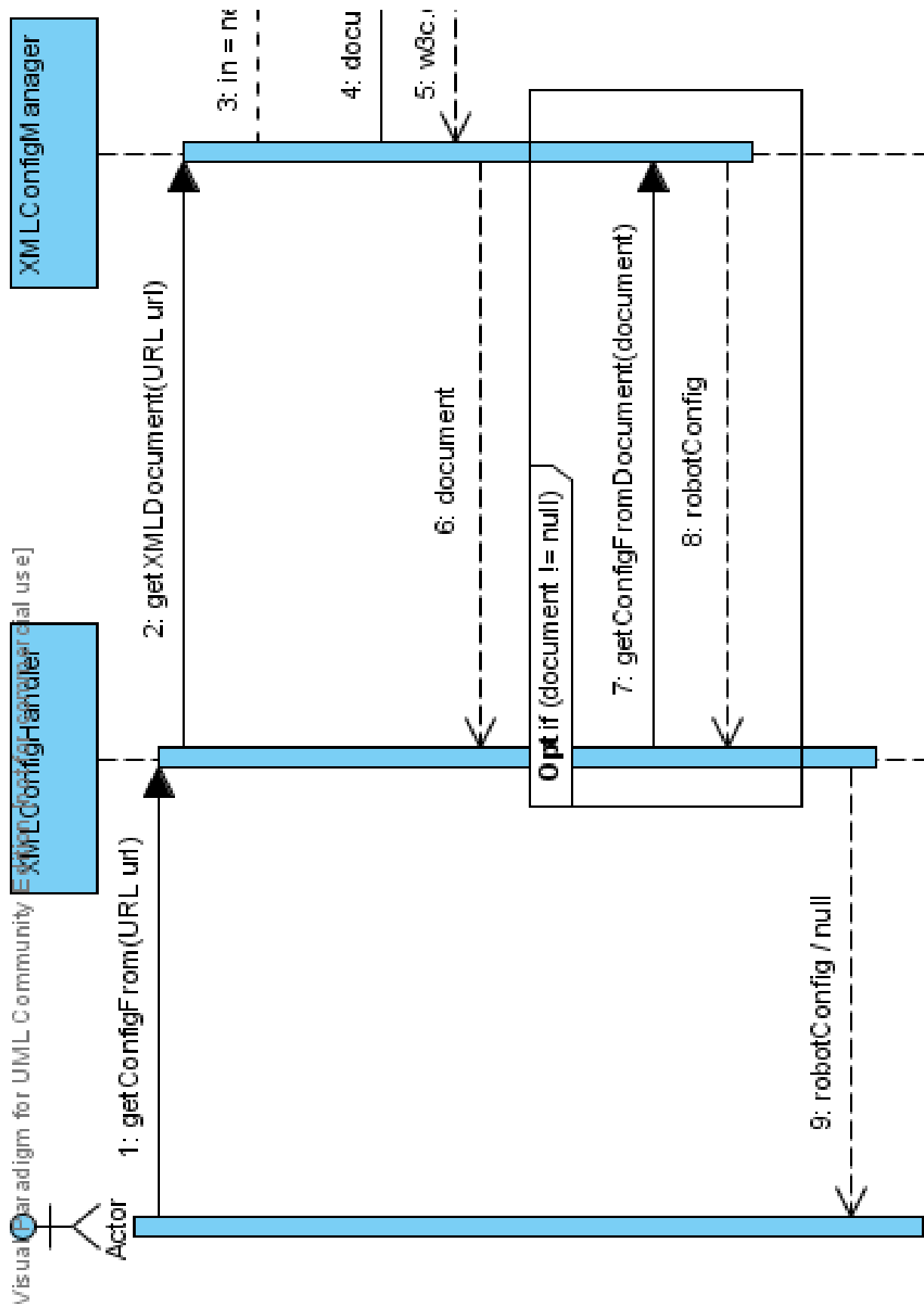


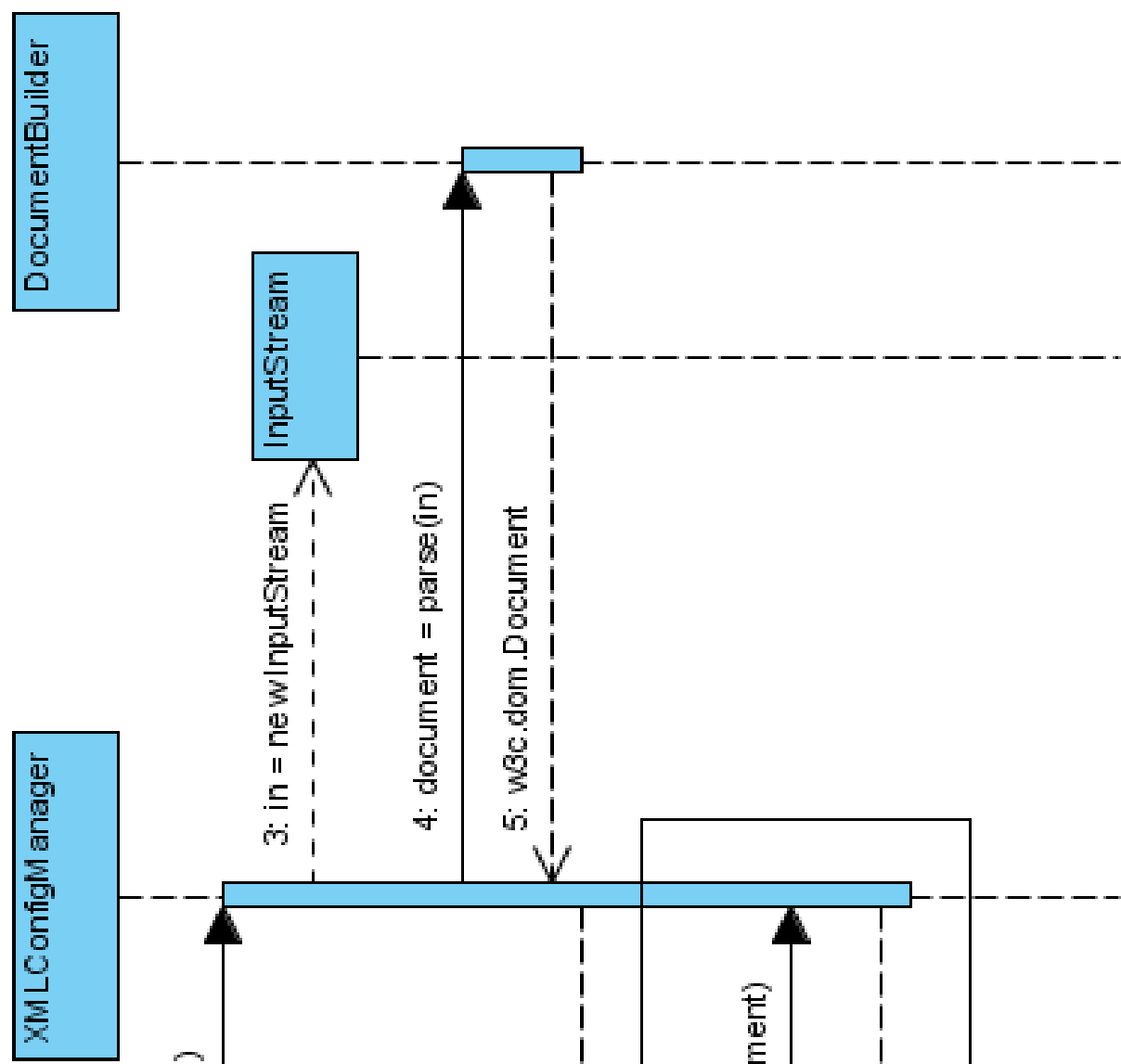




8.2 Release 2

8.2.1 Hent konfiguration fra XML fil





9. Brugergrænsefladetest

9.1 Think aloud test

Think-Aloud test

Disse skal udfyldes af systemudvikler mens bruger "tænker højt"

1. "Opret et nyt modul"

Tid	Start:	Slut:
Manglende funktionalitet	Ja*	Nej
Task Faliur	Ja*	Nej
Annoying	Ja*	Nej
Medium problem (efter lang tid)	Ja*	Nej
Minor problem (efter kort tid)	Ja*	Nej

*Skriv mere udførligt i noter

Noter:

2. "Se på SMR grafer i RHD modul"

Tid	Start:	Slut:
Manglende funktionalitet	Ja*	Nej
Task Faliur	Ja*	Nej
Annoying	Ja*	Nej
Medium problem (efter lang tid)	Ja*	Nej
Minor problem (efter kort tid)	Ja*	Nej

*Skriv mere udførligt i noter

Noter:

3. "Send en kommando "var allcopy" til RHD modulet"

Tid	Start:	Slut:
Manglende funktionalitet	Ja*	Nej
Task Faliur	Ja*	Nej
Annoying	Ja*	Nej
Medium problem (efter lang tid)	Ja*	Nej
Minor problem (efter kort tid)	Ja*	Nej

*Skriv mere udførligt i noter

Noter:

4. "Se Variabletræ for RHD modulet"

Tid	Start:	Slut:
Manglende funktionalitet	Ja*	Nej
Task Faliur	Ja*	Nej
Annoying	Ja*	Nej
Medium problem (efter lang tid)	Ja*	Nej
Minor problem (efter kort tid)	Ja*	Nej

*Skriv mere udførligt i noter

Noter:

9.2 Spørgeskema

Spørgeskema ud fra brugervenlighedsfaktorer

Dette udfyldes af bruger efter de har udført think-aloud test.

Fit for use

1. Hvor nemt var det at udføre de handlinger du blev bedt om?
Sæt kryds (1 = meget svært, 5 = meget nemt)

1	2	3	4	5
---	---	---	---	---

Ease of learning

2. Hvor nemt var det at lærer at bruge MARG?
Sæt kryds (1 = meget svært, 5 = meget nemt)

1	2	3	4	5
---	---	---	---	---

Task Efficiency

3. Hvor effektivt var det at bruge MARG?
Sæt kryds (1 = ikke effektivt , 5 = meget effektivt)

1	2	3	4	5
---	---	---	---	---

Ease of remembering

4. Hvis du kun skulle bruge systemet en gang imellem
hvor nemt ville det så være at huske hvordan man gør?
Sæt kryds (1 = meget svært, 5 = meget nemt)

1	2	3	4	5
---	---	---	---	---

Subjective satisfaction

5. Hvor tilfreds er du med systemet?
Sæt kryds (1 = ikke tilfreds, 5 = meget tilfreds)

1	2	3	4	5
---	---	---	---	---

Understandability

6. Hvor nemt var det at forstå hvad systemet gjorde?
Sæt kryds (1 = meget svært, 5 = meget nemt)

1	2	3	4	5
---	---	---	---	---

10. Referat af DTU Møder

10.1 Release 1 d. 24/09/09

Vi startede mødet ud med at præsentere vores systems GUI. Vi viste de funktionaliteter som var blevet implementeret; Connection, RawData, Variabel træ. Projektpartner virkede tilfredse med GUI designet og i sær funktionaliteten med variabel træet. De savner grafisk visning for nogle specifikke variabler, så som IRSensor og LineSensor, hvilket vi nok må overveje at prioritere højere i planlægningen af kommende iterationer.

Vi havde en del diskussion og feedback i forhold til plugin strukturen og hvordan den skulle fungere. Der var nogle overvejelser omkring parsing af XML der indeholder binært data eller store mængder af data.

Efter mødet testede vi vores system på en SMR robot, hvor vi også kunne køre vores acceptance tests imod. Resultatet af vores acceptance tests var en succes. Vi testede en masse metodekald til MRC og RHD modulerne på SMR robotten, for at mappe interfacet af robotten og bedre kunne skrive acceptance tests i de næste iterationer.

Vi har aftalt at mødes igen torsdag d. 8 Oktober, hvor vi har vores Release 2.

Noter fra mødet:

- GUI
 - Visning af specifikke variabel data grafisk.
 - Linesensor, IRsensor, Motor
 - Se tegning for dette.
 - Træstruktur
 - Ændring af variabel værdi
 - F2 eller højreklik -> rediger?
 - Understøttelse af panel-plugin ++
 - STOP/Pause/Start knapper
 - space = pause? Eller anden genvej
 - Modul Status
 - STOR status felt ud for hver modul-knap i den venstre menu
 - Rød for ingen forbindelse.
 - Gul for forbindelse men ingen eller dårlig status.
 - Grøn for forbindelse og status ok.
 - Titel
 - Titel på browser-vindue med robottens navn?
 - Pop-up vindue ved start der siger "Velkommen til <robotnavn>"
 - Command field
 - Genbrug af tidligere sendte cmds inden for samme session.
 - Vha. Dropdown
- Backend
 - Subscribe vs. Central DB
 - Virker til at det bliver okay på denne måde, men skal have genbrug og tilgang af data i tankerne.

- XMLParsePlugin
 - tagContents(String contents). Hvad hvis binært/hex data?
 - Videre send ch[] i stedet for at lave det til String først
 - Størrelse af data modtaget? Skal det sendes af sted løbende eller når alt er modtaget?
 - Datatunge plugins/moduler skal måske bruge egen Client, så andre ikke overbelastes
- Plugin Struktur
 - 1. new Jpanel Form
 - 2. implements XMLParsePlugin, ModuleTabPlugin
 - doUpdate(), så den ikke behøver egen tråd.

Idéer til Release 3:

- Map Modul (Christian)
 - Hente mange variabler og koordinater og omsætte dem til et billede med Javas graphics metoder.
- Grafisk visning af specifikke variabler, se billede. **Ryk til Release 2?**
- Robottests: plugin hvor automatisk test kan køres for at tjekke robotens tilstand.
- Plugin eksempel. Et man kan kopiere og bygge videre på, slags skabelon
- Plugin dokumentation. Hvordan laves et plugin + lidt javadoc.

10.2 Release 2 d. 8/10/09

Referat af Release 2

Vi startede med at præsentere den opdaterede system GUI og den implementerede funktionalitet 'Vis grafer'. Projektpartner var meget tilfreds med 'Vis grafer for SMR variabler', for det grafiske betyder at systemet vil blive brugt. Vi havde ikke nået at implementere 'Juster SMR variabler' som har betydning for hjulenes værdier mht. hastighed. Vi havde også været lidt i tvivl omkring kravet til denne user-story. Projektpartner oplyste de detaljer der var nødvendige for at fastsætte kravet til user-story 'Juster SMR variabler'. Den videre præsentation af de implementerede funktionaliteter foregik ved at vi fik projektpartner til at udføre en think-aloud test. På den måde fik vi lavet en test af brugergrænsefladen og samtidig vist resten af de implementerede user-stories. Efterfølgende gennemgik vi noget af den implementerede kode. Projektpartner havde ingen indvendinger til koden. Projektpartner var generelt tilfreds med Release 2.

Vi aftale at mødes igen torsdag d. 22. oktober, hvor vi har Release 3.

Efter præsentationen gennemgik vi den foreløbige plan for Release 3 med projektpartner og fik prioriteret hvilke user-story der har højest forretningsværdi.

Vi etablerede også en forbindelse til en SMR robot og fik afprøvet 'Vis grafer for SMR variabler'. Det viste sig at vi skal have justeret graf værdier da det ikke var muligt at aflæse udsving tydeligt nok. Vi uploadede vores GUI til en SMR robot og tilgik efterfølgende SMR robotten. Det virkede efter hensigten og GUI startede automatisk. Ved tjek med tilgang fra andre computere på DTU viste det sig desværre at det ikke kunne lade sig gøre, måske pga. at Java version var forældet på de afprøvede computere. Projektpartner ville kontakte systemadministrator for en afklaring.

Vores system viste en alvorlig fejl under forbindelsen til SMR robot som fik systemet til at gå ned. Heldigvis fik vi lokaliseret fejlen som værende et overflow af data i raw data vinduet. Løsningen vil være at tilføje løbende sletning af data i raw data vinduet.

Imens vi havde forbindelse til en SMR robot fik vi også kørt vores acceptance test. Resultatet var en succes da alle acceptance test blev godkendt.

Noter fra mødet:

Evt. ekstra implementeringer:

- image plugin - til kamera som skal vise et image der som minimum kan fokuseres
- laser plugin - til laser scanner der som minimum viser om der leveres data(hex tal)
- Højtalere plugin – sende kommando
- Vis motor status – reset motor

Små GUI ændringer:

- visning af valgt modul knap fx som radiobutton
- Der må gerne stå encoder og speed på, ved hjulene
- variabeltræ bliver opdateret automatisk ved tryk på faneblad
- angivelse af tal 1-8 under hver sensor i 'Vis grafer for SMR variabler', max værdi for hver sensor er 255
- hjul kører med 2000 omdrejninger/min, skal angives som værdi hvor 128 er højest
- Vise anden fane ved start end Control.

10.3 Release 3 d. 29/10/09

Vi startede med at udlevere en cd-rom til projektpartner indeholdende alle systemfiler inklusiv java-doc og en guide forbeholdt de studerende på DTU på hvordan de kan implementere nye moduler og plugins. Vi gennemgik indholdet på cd-rom med uddybende forklaring til hvert emne. Derefter overførte projektpartner den nyeste version af systemet til SMR robot så vi kunne præsentere de nyeste implementeringer fra release 3. Projektpartner var tilfreds med resultatet og at systemet var brugbart for dem. De var meget tilfreds med den måde som sensorer og hjul blev vist på grafisk. Derudover var de godt tilfreds med at vi havde lavet det muligt at tilføje grænseværdier til udvalgte variabler.

Vi viste at systemet kunne afvikles på både en version af Windows og Linux. Endvidere at systemet kunne afvikles med forskellige browsere som Internet Explorer og Firefox.

Projektpartner etablerede forbindelse til en anden type robot og det virkede og variabeltræ blev vist som det skulle. Projektpartner havde dagen forinden testet om der kunne etableres forbindelse til en tredje type robot og det kunne der.

Projektpartner havde et ønske om at teste om kompilering af systemet kunne ske på deres Linux computer. Det kunne det ikke. Men fejlen bestod formentlig i at version af Netbeans eller Java SDK var forældet. Det samme var tilfældet når der blev forsøgt at etablere forbindelse til robot men denne gang var det Java som var forældet. Det var en Java version 1.5. Projektpartner ville teste om der kunne etableres en forbindelse til robot på deres Windows computer. Det kunne godt lade sig gøre. Her var Java min. version 1.6. Det samme var tilfældet da der blev forsøgt at etablere forbindelse fra deres bærbar med Windows.

Vi fik kørt vores acceptance test til release 3 og alle blev godkendt.

Afslutningsvis forhørte vi os muligheden for at låne en SMR robot til brug ved vores eksamen som en del af vores præsentation. Det kunne godt lade sig gøre men med forbehold for hvilken ugedag det kommer til at dreje sig omkring. Projektpartner ville også høre om det var muligt for dem at overvære vores eksamen. Vi aftalte at vende tilbage med nærmere info efter afklaring med vejleder.

11. GUIDE: Creating plugins for MARG

11.1 Creating a plugin to parse XML

Creating the Plugin

To create a plugin that can be used to parse received XML data, you must create a new class that implements `XMLParsePlugin`. It's suggested to place the new class inside the package **marg.model.plugin**.

```
public class SomeDataPlugin implements XMLParsePlugin {  
    ...  
}
```

The interface `XMLParsePlugin`, which should be implemented, is defined as follows:

Method Summary	
void	<code>endElement</code> (java.lang.String tagName) Called when the endtag of an element is received
void	<code>setPropertyChangeSupport</code> (java.beans.PropertyChangeSupport prop) Sets the <code>PropertyChangeSupport</code> object of the parent <code>XMLParser</code> .
void	<code>startElement</code> (java.lang.String tagName, org.xml.sax.Attributes atts) Called when the starttag of an element is received
void	<code>tagContents</code> (java.lang.String parentTag, char[] ch, int start, int length) Called when contents of an element is received.

Further information about each method and how your plugin should implement it can be seen in the JavaDoc for `XMLParsePlugin`.

`setPropertyChangeSupport` is used to retrieve the central `PropertyChangeSupport` object. This object is used to subscribe to and perform callbacks to all subscribers when new data has been received. The received object should be stored in a local variable so that it can be used in other methods. Further information about each method and what it does can be found in the JavaDoc of `XMLParsePlugin`.

Firing a callback event

To fire a callback from your plugin, use this example for guidance:

```
public class SomeDataPlugin implements XMLParsePlugin {  
  
    public final static String SUBSCRIBE_DATANAME = "uniqueString";  
    ...  
    public void someMethod {  
        prop.firePropertyChange(SUBSCRIBE_DATANAME, oldValue, newValue);  
    }  
}
```

The reason it's a good idea to use a String that's public, can be seen from the following example of a plugin subscribing to callbacks of this type of data:

```
handler.addPropertyChangeListener(this, SomeDataPlugin.SUBSCRIBE_DATANAME);
```

As you can see there is no need to use an exact String. It's possible to just refer to the class' public variable.

This makes it easier to change the String without having to correct it multiple places and there's no need to worry about spelling it wrong. This is just a suggested way to do callbacks; it is not in any way mandatory.

Receiving callbacks

Any class that wishes to receive callbacks when `firePropertyChange` is called in an `XMLParsePlugin` needs to subscribe to the particular property and provide a `PropertyChangeListener` object to receive the callbacks.

Here are examples of how it is done for each module in the GUI:

```
public class ModuleTabs extends JPanel implements RobotModule,
PropertyChangeListener {

    ...
    handler.addPropertyChangeListener(this, RawDataPlugin.SUBSCRIBE_RAWDATA,
VarDataPlugin.SUBSCRIBE_VARDATA);
    ...

    public void propertyChange(PropertyChangeEvent evt) {
        if (evt.getPropertyName().equals(RawDataPlugin.SUBSCRIBE_RAWDATA)) {
            newRawData(evt);
        } else if (evt.getPropertyName().equals(VarDataPlugin.SUBSCRIBE_VARDATA)) {
            newVarData(evt);
        }
    }
}
```

As seen in the code above the module has subscribed to events with itself as a listener. Therefore it has implemented the interface `PropertyChangeListener`, which defines the method *propertyChange*. Since the module is receiving events from several plugins, it first separates the event according to which `propertyName` it has and then sends it on.

Adding your new XMLParsePlugin to the XMLParser

In order for your `XMLParsePlugin` to start receiving data, you must add it to the `XMLParser` of a client. This is done through its handler like this:

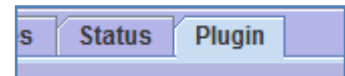
```
handler.getXMLParser().addParsePlugin(new SomeDataPlugin());
```

This will only register the plugin for this particular `XMLParser`. Each module in the system has its own `XMLParser`. To add a new parser for every single module that gets added, see the `initXMLClient` method in the `ModuleTabs` class.

11.2 Creating a plugin for a module

Creating the plugin

In order to create a plugin for a module you need to create a new class that implements the `ModulePlugin` interface. The new class **must** be created inside the package **marg.gui.plugin** in order to be automatically found when MARG is run.



```
public class ClassName implements ModulePlugin {
    ...
}
```

The interface `ModulePlugin`, which should be implemented, is defined as follows:

Method Summary	
void	<code>doUpdate()</code> Called by parent module regularly every second by default.
<code>javax.swing.JPanel</code>	<code>getJPanel()</code> Return the visual representation of your plugin here.
<code>java.lang.String</code>	<code>getPluginName()</code> Gets the name of your plugin.
void	<code>setXMLClientHandler(XMLClientHandler handler)</code> Offers the handler to your plugin.
void	<code>startPlugin()</code> Called by the parent module to start a plugin.
void	<code>stopPlugin()</code> Called by the parent module during cleanup process when the module or entire applet is closed.

Further detailed information about each method and what it does can be found in the JavaDoc of `ModulePlugin`.

A module plugin can do pretty much anything you like once it has implemented the basic interface. Most often you'll want to subscribe to some data changes and make a visual representation of these on your `JPanel`. The `JPanel` can be a separate class or it can even be the `ModulePlugin` itself. Should it be needed by your plugin, it is possible to send commands through the `sendCmd` method of the handler.

Creating a `JPanel` that implements both plugin interfaces

With our design it is possible to gather all of the plugins and needed classes in a single Class like this:

```
public class MyPlugin extends JPanel implements ModulePlugin, XMLParsePlugin {

    private XMLClientHandler handler;
    private PropertyChangeSupport prop;

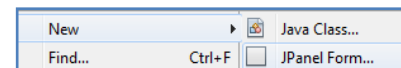
    public void setXMLClientHandler(XMLClientHandler handler) {
        this.handler = handler;
    }
    public void setPropertyChangeSupport(PropertyChangeSupport prop) {
        this.prop = prop;
    }

    public JPanel getJPanel() {
        return this;
    }

    public String getPluginName() {
        return "MyPlugin";
    }

    public void startPlugin() {
        //Registers itself as an XMLParsePlugin
        handler.addParsePlugin(this);
    }
    (Left out rest of the implementation)
}
```

This way you can even create a plugin where you can design your JPanel with drag&drop, and implement the functionality behind it, in the very same class. In the popular Java IDE; Netbeans, you would do this by right-clicking in your project, choosing New and then "JPanel Form...". Then you would just need to add the 2 interfaces as seen above.

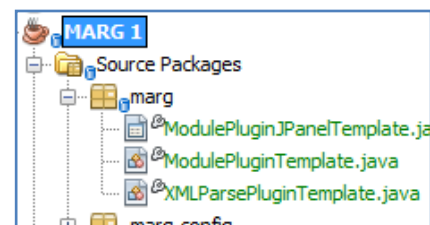


11.3 Starting from Templates


To make it easier to create new plugins we have created a few templates, which can be found inside the source of our project. When developing new plugins we highly recommend using Netbeans and opening our project folder.

All templates can be found in the base **marg** package.


When creating a plugin from a template, a copy of the template should be moved to its appropriate package and the template renamed to reflect the new plugin.



Templates to do with ModulePlugin should be copied to package:

 **marg.gui.plugin**

Templates to do with XMLParsePlugin should be copied to package:

 **marg.model.plugin**

Templates are placed here so that the system doesn't offer the templates as actual plugins in the GUI.

12. Projekt Kontrakt

Projekt Kontrakt

Denne kontrakt er gældende imellem følgende parter:

Projektgruppen DTU Elektro

Definition af opgave

Projektgruppen og projektpartner har sammensat følgende projektoplæg:

- Der skal laves en brugergrænseflade til en robot
- Det skal være en Java applet der kan afvikles i hvilken som helst internet browser.
- Det skal være modulært og konfigurerbart så det kan bruges til flere robotter og/eller nye moduler.
- Det skal bruge en socket-baseret TCP/IP forbindelse til at kommunikere med robotten.
- Det skal både kunne sende og modtage data fra robotten i XML.
- Det skal være nemt at lave et nyt specialiseret modul ud fra et generelt modul. Dokumentation til hvordan dette gøres er vigtig.
- Denne skal have en standardiseret måde at vise alle variable på. Træstruktur er fortrukket.
- Der skal være et vindue for forbindelsen der viser det rå data der kommer fra robotten.
 - Det skal være muligt at sende kommandoer til robotten fra dette vindue.
- Det skal hele tiden være muligt at se de status variabler brugeren ønsker at overvåge i GUI på et givet tidspunkt.

Betingelser

Periode

Projektet løber over perioden 24.8.2009 til 5.11.2009 og har til overordnet formål at danne grundlag for projektgruppens afsluttende eksamen på datamatikeruddannelsen.

Ressourcer

Projektpartner stiller faciliteter til rådighed i projektperioden der gør det muligt for projektgruppen at kommunikere med tidligere nævnte robotter samt gør det muligt at teste udviklet software.

Ophavsret

Projektgruppen har alle rettigheder til udviklet projektrapport. Der udleveres en kopi af projektrapporten til projektpartner. Hele eller dele af projektrapporten må kun publiceres efter aftale med projektgruppen.

Software er underlagt LGPL licens

Honorering

Projektgruppen har ingen krav på honorering fra projektpartner for tidsforbruget brugt på projektet i projektperioden.

Gyldigheden af denne kontrakt er betinget af projektvejleders godkendelse af kontraktens indhold.

Projektgruppen underskrift, dato / -2009

Bjørn Truelsen

Daniel Freiling

Kristina Hansen

Projektvejleder underskrift, dato / -2009

Michael Claudius

Projektpartner underskrift, dato / -2009

Kode Bilag

Indhold

marg.config.ControlButton.....	2
marg.config.Module	3
marg.config.Robot	5
marg.config.XMLConfigHandler	7
marg.config.XMLConfigManager	8
marg.gui.dialogs.AddModuleDialog	11
marg.gui.dialogs.AddStatusLimitDialog	13
marg.gui.dialogs.MonitorVariableDialog	16
marg.gui.MainGUIHandler	18
marg.gui.MargGUI.....	20
marg.gui.ModuleMenu	23
marg.gui.ModuleTabs	28
marg.gui.RobotModule	34
marg.gui.VariableTree.....	35
marg.gui.plugin.ModulePlugin	37
marg.gui.plugin.SMRPlugin	38
marg.gui.plugin.smr.LineGraph.....	41
marg.gui.plugin.smr.RobotWheel	43
marg.gui.plugin.smr.WheelControl	45
marg.handlers.XMLClientHandler	47
marg.images.ResUtils.....	48
marg.model.AbstractModuleClient.....	49
marg.model.ModuleClient	50
marg.model.ModuleVariable	51
marg.model.MonitoredVariable.....	52
marg.model.StatusVariable	53
marg.model.ValueLimit	54
marg.model.XMLClient	55
marg.model.XMLClientGUI	56
marg.model.XMLParser	57
marg.model.plugin.XMLParsePlugin	59
marg.model.plugin.RawDataPlugin	60
marg.model.plugin.VarDataPlugin	61
marg.test.mockserver.JUnitMockServer	62
marg.test.mockserver.MockAllCopyServer	63
marg.test.mockserver.MockServer	65
marg.test.mockserver.MockServerLineIR.....	67
marg.test.mockserver.MockServerWheels	69
marg.util.Log	71
marg.util.PluginManager	72
marg.XMLParsePluginTemplate	75
marg.ModulePluginTemplate	76
marg.ModulePluginJPanelTemplate	77
JUnit MARGTestSuite	78
JUnit MonitoredVariableTest	79
JUnit ValueLimitTest	81
JUnit XMLClientTest.....	83

marg.config.ControlButton

```
06 package marg.config;
07
08 /**
09  *
10  * @author MARG MARG
11  */
12 public class ControlButton {
13
14     private String moduleName;
15     private String command;
16
17     public ControlButton(String moduleName, String command) {
18         this.moduleName = moduleName;
19         this.command = command;
20     }
21
22     public String getCommand() {
23         return command;
24     }
25
26     public String getModuleName() {
27         return moduleName;
28     }
29 }
```

marg.config.Module

```
006 package marg.config;
007
008 import java.util.ArrayList;
009
010 /**
011  *
012  * @author MARG Daniel
013  */
014 public class Module {
015
016     public Module(String name, String host, int port) {
017         this.port = port;
018         this.host = host;
019         this.name = name;
020     }
021     private ArrayList<String> modulePlugins = new ArrayList<String>();
022
023     public void addPlugin(String name) {
024         if (name != null && !name.equals("")) {
025             modulePlugins.add(name);
026         }
027     }
028
029     public void removePlugin(String name) {
030         modulePlugins.remove(name);
031     }
032
033     public ArrayList<String> getModulePlugins() {
034         return modulePlugins;
035     }
036     private boolean autoConnect = false;
037
038     public boolean isAutoConnect() {
039         return autoConnect;
040     }
041
042     public void setAutoConnect(boolean autoConnect) {
043         this.autoConnect = autoConnect;
044     }
045     private int port = 0;
046
047     /**
048      * Get the value of port
049      *
050      * @return the value of port
051      */
052     public int getPort() {
053         return port;
054     }
055
056     /**
057      * Set the value of port
058      *
059      * @param port new value of port
060      */
061     public void setPort(int port) {
062         this.port = port;
063     }
064     private String host = "";
065
066     /**
067      * Get the value of moduleHost
```

```

068      *
069      * @return the value of moduleHost
070      */
071      public String getModuleHost() {
072          return host;
073      }
074
075      /**
076       * Set the value of moduleHost
077       *
078       * @param moduleHost new value of moduleHost
079       */
080      public void setModuleHost(String moduleHost) {
081          this.host = moduleHost;
082      }
083      private String name = "";
084
085      /**
086       * Get the value of moduleName
087       *
088       * @return the value of moduleName
089       */
090      public String getModuleName() {
091          return name;
092      }
093
094      /**
095       * Set the value of moduleName
096       *
097       * @param moduleName new value of moduleName
098       */
099      public void setModuleName(String moduleName) {
100          this.name = moduleName;
101      }
102
103      @Override
104      public String toString() {
105          StringBuilder sb = new StringBuilder();
106          sb.append(name + ", " + host + ":" + port);
107          sb.append(", Auto-Connect: " + autoConnect);
108          for (String s : modulePlugins) {
109              sb.append(", Plugin: " + s);
110          }
111          return sb.toString();
112      }
113
114  }

```

marg.config.Robot

```
06 package marg.config;
07
08 import java.util.ArrayList;
09 import java.util.List;
10
11 /**
12  * @author MARG
13  */
14 public class Robot {
15
16     private List<Module> modules = new ArrayList<Module>();
17     private ControlButton startButton;
18     private ControlButton pauseButton;
19     private ControlButton stopButton;
20     private String robotName;
21
22     public Robot() {
23     }
24
25     public boolean removeModule(Module module) {
26         return modules.remove(module);
27     }
28
29     public boolean containsModule(Module module) {
30         return modules.contains(module);
31     }
32
33     public boolean addModule(Module module) {
34         return modules.add(module);
35     }
36
37     public List<Module> getModules() {
38         return modules;
39     }
40
41     public ControlButton getPauseButton() {
42         return pauseButton;
43     }
44
45     public void setPauseButton(ControlButton pauseButton) {
46         this.pauseButton = pauseButton;
47     }
48
49     public ControlButton getStartButton() {
50         return startButton;
51     }
52
53     public void setStartButton(ControlButton startButton) {
54         this.startButton = startButton;
55     }
56
57     public ControlButton getStopButton() {
58         return stopButton;
59     }
60
61     public void setStopButton(ControlButton stopButton) {
62         this.stopButton = stopButton;
63     }
64
65     public void setRobotName(String robotName) {
66         this.robotName = robotName;
67     }
68 }
```

```
69
70     public String getRobotName() {
71         return robotName;
72     }
76 }
```

marg.config.XMLConfigHandler

```
05 package marg.config;
06
07 import java.net.MalformedURLException;
08 import java.net.URL;
09 import java.util.logging.Level;
10 import java.util.logging.Logger;
11 import marg.gui.MargGUI;
12
13 /**
14  *
15  * @author MARG
16  */
17 public class XMLConfigHandler {
18
19     private XMLConfigManager configMan;
20
21     public XMLConfigHandler() {
22         configMan = new XMLConfigManager();
23     }
24
25     public Robot getConfigFrom(URL url) {
26         org.w3c.dom.Document doc = configMan.getXMLDocument(url);
27         Robot loadedRobot = configMan.getConfigFromDocument(doc);
28         return loadedRobot;
29     }
30
31     public static boolean isLocalURL(URL url) {
32         if (url.getProtocol().equals("file"))
33             return true;
34         else
35             return false;
36     }
37
38     public static URL getFileURL(URL baseURL, String fileName) {
39         try {
40             URL fileURL = new URL(baseURL, fileName);
41             return fileURL;
42         } catch (MalformedURLException ex) {
43             Logger.getLogger(XMLConfigHandler.class.getName()).log(Level.SEVERE, null, ex);
44             return baseURL;
45         }
46     }
47
48     public static URL fixLocalUrl(URL url) {
49         String protocol = url.getProtocol();
50         if (protocol.equals("file")) {
51             String path = url.getPath();
52             try {
53                 int buildOffset = path.indexOf("build");
54                 int distOffset = path.indexOf("dist");
55                 if (buildOffset != -1) { //We have a build-folder match
56                     url = new URL(protocol + ":" + path.substring(0, buildOffset)
57 );
58                 } else if (distOffset != -1) { //we have a dist-folder match;
59                     url = new URL(protocol + ":" + path.substring(0, distOffset)
60 );
61                 }
62             } catch (MalformedURLException ex) {
63                 Logger.getLogger(MargGUI.class.getName()).log(Level.SEVERE, null, ex);
64             }
65         }
66     }
```

```

63     }
64     System.out.println("Fixed Local URL: " + url);
65     return url;
66 }
67 }

```

marg.config.XMLConfigManager

```

005 package marg.config;
006
007 import java.io.File;
008 import java.io.FileNotFoundException;
009 import java.io.IOException;
010 import java.io.InputStream;
011 import java.net.URL;
012 import java.util.ArrayList;
013 import java.util.logging.Level;
014 import java.util.logging.Logger;
015 import javax.xml.parsers.DocumentBuilder;
016 import javax.xml.parsers.DocumentBuilderFactory;
017 import javax.xml.parsers.ParserConfigurationException;
018 import org.w3c.dom.Document;
019 import org.w3c.dom.Element;
020 import org.w3c.dom.NamedNodeMap;
021 import org.w3c.dom.Node;
022 import org.w3c.dom.NodeList;
023 import org.xml.sax.SAXException;
024
025 /**
026  *
027  * @author MARG
028  */
029 public class XMLConfigManager {
030
031     private Logger log = Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);
032     private Robot robot;
033
034     public Document getXMLDocument(URL url) {
035         Document doc = null;
036         try {
037             InputStream in = url.openConnection().getInputStream();
038             DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
039             DocumentBuilder builder = factory.newDocumentBuilder();
040             doc = builder.parse(in);
041         } catch (FileNotFoundException ex) {
042             log.warning("File not found for url: " + url);
043         } catch (SAXException ex) {
044             Logger.getLogger(XMLConfigManager.class.getName()).log(Level.SEVERE, null, ex);
045         } catch (ParserConfigurationException ex) {
046             Logger.getLogger(XMLConfigManager.class.getName()).log(Level.SEVERE, null, ex);
047         } catch (IOException ex) {
048             Logger.getLogger(XMLConfigManager.class.getName()).log(Level.SEVERE, null, ex);
049         }
050         return doc;
051     }
052
053     //TODO: I think this could use some refactoring perhaps? :D
054     public Robot getConfigFromDocument(Document document) {
055         robot = new Robot();
056         if (document != null) {

```



```

057         //<robot>
058         NodeList robotElements = document.getElementsByTagName("robot");
059         Element robotNode = (Element) robotElements.item(0);
060         if (robotNode != null) {
061             //<name> [1]
062             NodeList nameNodes = robotNode.getElementsByTagName("name");
063             if (nameNodes.getLength() > 0) {
064                 Node nameNode = nameNodes.item(0).getChildNodes().item(0);
065                 String name = getNodeValue(nameNode);
066                 robot.setRobotName(name);
067             }
068
069             //<controlButtons> [0..1]
070             NodeList ctrlBtnNodes = robotNode.getElementsByTagName("contro
lButtons");
071             if (ctrlBtnNodes.getLength() > 0) {
072                 Element ctrlBtnNode = (Element) ctrlBtnNodes.item(0);
073                 Node startNode = ctrlBtnNode.getElementsByTagName("start"
.item(0);
074                 Node pauseNode = ctrlBtnNode.getElementsByTagName("pause")
.item(0);
075                 Node stopNode = ctrlBtnNode.getElementsByTagName("stop").i
tem(0);
076
077                 String startModule = getAttributeValueFor(startNode, "modu
leName");
078                 String startCmd = getAttributeValueFor(startNode, "command
");
079                 ControlButton startButton = new ControlButton(startModule,
startCmd);
080                 String pauseModule = getAttributeValueFor(pauseNode, "modu
leName");
081                 String pauseCmd = getAttributeValueFor(pauseNode, "command
");
082                 ControlButton pauseButton = new ControlButton(pauseModule,
pauseCmd);
083                 String stopModule = getAttributeValueFor(stopNode, "module
Name");
084                 String stopCmd = getAttributeValueFor(stopNode, "command")
;
085                 ControlButton stopButton = new ControlButton(stopModule, s
topCmd);
086                 robot.setStartButton(startButton);
087                 robot.setPauseButton(pauseButton);
088                 robot.setStopButton(stopButton);
089             }
090
091             //<module> [1..*]
092             NodeList moduleNodes = robotNode.getElementsByTagName("module"
);
093             for (int i = 0; i < moduleNodes.getLength(); i++) {
094                 Element moduleNode = (Element) moduleNodes.item(i);
095                 if (moduleNode != null) {
096                     //<name> [1]
097                     Node nameNode = moduleNode.getElementsByTagName("name"
).item(0).getChildNodes().item(0);
098                     String name = getNodeValue(nameNode);
099                     //<host> [0..1]
100                     Node hostNode = moduleNode.getElementsByTagName("host"
).item(0).getChildNodes().item(0);
101                     String host = getNodeValue(hostNode);
102                     //<port> [0..1]
103                     Node portNode = moduleNode.getElementsByTagName("port"
).item(0).getChildNodes().item(0);

```

```

104         String portString = getNodeValue(portNode);
105         Module newModule = null;
106         try {
107             if (name.equals("")) {
108                 throw new IllegalArgumentException("XML Module
109 " + (i + 1) + " had no name and could therefore not be added");
110             }
111             int port = 0;
112             try {
113                 port = Integer.parseInt(portString);
114             } catch (NumberFormatException ex) {
115             }
116             newModule = new Module(name, host, port);
117         } catch (IllegalArgumentException ex) {
118             log.info(ex.getMessage());
119         }
120         if (newModule != null) {
121             //<plugin> [0..*]
122             NodeList pluginNodes = moduleNode.getElementsByTag
123 Name("plugin");
124             if (pluginNodes.getLength() > 0) {
125                 for (int j = 0; j < pluginNodes.getLength(); j
126 +++) {
127                     Node pluginNode = pluginNodes.item(j).getC
128 hildNodes().item(0);
129                     String plugin = getNodeValue(pluginNode);
130                     newModule.addPlugin(plugin);
131                 }
132             }
133             //<autoConnect> [0..1]
134             NodeList autoConnectNodes = moduleNode.getElements
135 ByTagName("autoConnect");
136             if (autoConnectNodes.getLength() > 0) {
137                 Node autoConnectNode = moduleNode.getElementsB
138 yTagName("autoConnect").item(0).getChildNodes().item(0);
139                 String autoConnectString = getNodeValue(autoCo
140 nnectNode);
141                 boolean autoConnect = false;
142                 if (autoConnectString.equalsIgnoreCase("true")
143 ) {
144                     autoConnect = true;
145                 }
146                 newModule.setAutoConnect(autoConnect);
147             }
148             //Done, now add it to the arraylist
149             robot.addModule(newModule);
150             log.info("Parsed module from XML: " + newModule);
151         }
152     }
153 }
154
155 private String getAttributeValueFor(Node node, String attName) {
156     NamedNodeMap atts = node.getAttributes();
157     if (atts != null) {
158         Node attribNode = atts.getNamedItem(attName);
159         if (attribNode != null) {
160             return getNodeValue(attribNode.getChildNodes().item(0));
161         }
162     }
163 }

```

```

160         return "";
161     }
162
163     private String getNodeValue(Node node) {
164         if (node != null) {
165             return node.getNodeValue();
166         } else {
167             return "";
168         }
169     }
170
171     public static void main(String[] args) throws Exception {
172         XMLConfigManager config = new XMLConfigManager();
173         //URL url = config.getClass().getClassLoader().getResource("robot.xml"
174     );
175         URL url = new File("robot.xml").toURI().toURL();
176         Document doc = config.getXMLDocument(url);
177         config.getConfigFromDocument(doc);
178     }

```

marg.gui.dialogs.AddModuleDialog

```

012 package marg.gui.dialogs;
013
014 /**
015  *
016  * @author MARG
017  */
018 public class AddModuleDialog extends javax.swing.JDialog {
019
020     private String host;
021     private String moduleName;
022     private int port = 0;
023     private boolean autoConnect;
024     private boolean wasApproved = false;
025
026     /** Creates new form NewModuleDialog */
027     public AddModuleDialog(java.awt.Frame parent, boolean modal) {
028         super(parent, modal);
029         initComponents();
030         this.getRootPane().setDefaultButton(jButton1);
031     }
032
033     public String getHost() {
034         return host;
035     }
036
037     public boolean wasApproved() {
038         return wasApproved;
039     }
040
041     public int getPort() {
042         return port;
043     }
044
045     public String getModuleName() {
046         return moduleName;
047     }
048
049     public boolean getAutoConnect() {
050         return autoConnect;
051     }

```

```

052
053     /** This method is called from within the constructor to
054     * initialize the form.
055     * WARNING: Do NOT modify this code. The content of this method is
056     * always regenerated by the Form Editor.
057     */
058     @SuppressWarnings("unchecked")
059     // <editor-fold defaultstate="collapsed" desc="Generated Code"> // GEN-
BEGIN: initComponents
060     private void initComponents() {
061         // Removed auto-generated code
243
244     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { //GE
N-FIRST:event_jButton1ActionPerformed
245         wasApproved = true;
246         moduleName = jTextName.getText();
247         host = jTextHost.getText();
248         autoConnect = jCheckBox1.isSelected();
249         try {
250             port = Integer.parseInt(jTextPort.getText());
251             this.setVisible(false);
252         } catch (NumberFormatException ex) {
253             System.err.println("Could not read number from input: " + jTextPort
.getText());
254         }
255     } // GEN-LAST:event_jButton1ActionPerformed
256
257     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) { //GE
N-FIRST:event_jButton2ActionPerformed
258         this.setVisible(false);
259     } // GEN-LAST:event_jButton2ActionPerformed
260
261     /**
262     * @param args the command line arguments
263     */
264     public static void main(String args[]) {
265         java.awt.EventQueue.invokeLater(new Runnable() {
266             public void run() {
267                 AddModuleDialog dialog = new AddModuleDialog(new javax.swing.J
Frame(), true);
268                 dialog.addWindowListener(new java.awt.event.WindowAdapter() {
269                     public void windowClosing(java.awt.event.WindowEvent e) {
270                         System.exit(0);
271                     }
272                 });
273                 dialog.setVisible(true);
274             }
275         });
276     }
277
278     // Variables declaration - do not modify // GEN-BEGIN:variables
279     // Removed auto-generated code
296
297 }

```

marg.gui.dialogs.AddStatusLimitDialog

```
011 package marg.gui.dialogs;
012
013 import javax.swing.JOptionPane;
014 import marg.model.ModuleVariable;
015 import marg.model.StatusVariable;
016 import marg.model.ValueLimit;
017
018 /**
019  *
020  * @author MARG
021  */
022 public class AddStatusLimitDialog extends javax.swing.JDialog {
023
024     private boolean wasApproved = false;
025     private ValueLimit valueLimit;
026
027     public AddStatusLimitDialog(java.awt.Frame parent, boolean modal, StatusVa
riable statVar) {
028         super(parent, modal);
029         initComponents();
030         ValueLimit oldLimit = statVar.getLimit();
031         if (oldLimit == null) {
032             String varValue = statVar.getModuleVar().getValue();
033             initDialogFields(varValue, varValue, statVar.getModuleVar().getVar
Name());
034         } else {
035             initDialogFields(String.valueOf(oldLimit.getMinValue()), String.va
lueOf(oldLimit.getMaxValue()), statVar.getModuleVar().getVarName());
036             initComparisonSigns(statVar.getLimit());
037         }
038     }
039
040     private void initDialogFields(String minText, String maxText, String varNa
me) {
041         jTextMin.setText(minText);
042         jTextMax.setText(maxText);
043         jLabelVar.setText(jLabelVar.getText() + varName);
044     }
045
046     private void initComparisonSigns(ValueLimit limit) {
047         if (limit.getMinimumComparison() == ValueLimit.LESS_THAN) {
048             jComboMinComp.setSelectedItem("<");
049         }
050         if (limit.getMaximumComparison() == ValueLimit.LESS_THAN) {
051             jComboMaxComp.setSelectedItem("<");
052         }
053     }
054
055     public boolean wasApproved() {
056         return wasApproved;
057     }
058
059     public ValueLimit getValueLimit() {
060         return valueLimit;
061     }
062
063     /** This method is called from within the constructor to
064      * initialize the form.
065      * WARNING: Do NOT modify this code. The content of this method is
066      * always regenerated by the Form Editor.
067      */
068     @SuppressWarnings("unchecked")
```

```

069      // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN: initComponents
070      private void initComponents() {
213      } // Removed auto-generated code
214
215      private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { //GEN-
N-FIRST:event_jButton1ActionPerformed
216          String minText = jTextMin.getText();
217          String maxText = jTextMax.getText();
218          try {
219              Double min = Double.parseDouble(minText);
220              Double max = Double.parseDouble(maxText);
221              if (min > max) {
222                  throw new NumberFormatException();
223              }
224              String minCompString = (String) jComboMinComp.getSelectedItem();
225              String maxCompString = (String) jComboMaxComp.getSelectedItem();
226              int maxComparison;
227              int minComparison;
228              if (minCompString.equals("@%")) {
229                  minComparison = ValueLimit.LESS_THAN_EQUALS;
230              } else {
231                  minComparison = ValueLimit.LESS_THAN;
232              }
233              if (maxCompString.equals("@%")) {
234                  maxComparison = ValueLimit.LESS_THAN_EQUALS;
235              } else {
236                  maxComparison = ValueLimit.LESS_THAN;
237              }
238              valueLimit = new ValueLimit(min, minComparison, max, maxComparison
);
239              wasApproved = true;
240              this.setVisible(false);
241          } catch (NumberFormatException ex) {
242              JOptionPane.showMessageDialog(this, "Min or Max error", "Parse Err
or", JOptionPane.ERROR_MESSAGE);
243          }
244      } //GEN-LAST:event_jButton1ActionPerformed
245
246      private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) { //GEN-
N-FIRST:event_jButton2ActionPerformed
247          this.setVisible(false);
248      } //GEN-LAST:event_jButton2ActionPerformed
249
250      /**
251       * @param args the command line arguments
252       */
253      public static void main(String args[]) {
254          java.awt.EventQueue.invokeLater(new Runnable() {
255
256              private StatusVariable statVar = new StatusVariable(new ModuleVari
able("core.version", "d", "202"));
257
258              public void run() {
259                  AddStatusLimitDialog dialog = new AddStatusLimitDialog(new jav
ax.swing.JFrame(), true, statVar);
260                  dialog.addWindowListener(new java.awt.event.WindowAdapter() {
261
262                      public void windowClosing(java.awt.event.WindowEvent e) {
263                          System.exit(0);
264                      }
265                  });
266                  dialog.setVisible(true);
267

```

```
268         });  
269     }  
270     // Variables declaration - do not modify//GEN-BEGIN:variables  
271     // Removed auto-generated code  
285 }
```

marg.gui.dialogs.MonitorVariableDialog

```
011 package marg.gui.dialogs;
012
013 import javax.swing.JOptionPane;
014 import marg.model.ModuleVariable;
015 import marg.model.MonitoredVariable;
016
017 /**
018  *
019  * @author MARG
020  */
021 public class MonitorVariableDialog extends javax.swing.JDialog {
022
023     private boolean wasApproved = false;
024     private MonitoredVariable monitoredVar;
025     private ModuleVariable modVar;
026
027     public MonitorVariableDialog(java.awt.Frame parent, boolean modal, ModuleV
028     ariable modVar) {
029         super(parent, modal);
030         initComponents();
031         this.modVar = modVar;
032         String varName = modVar.getShortVarName();
033         varName = varName.substring(0, 1).toUpperCase() + varName.substring(1,
034         varName.length());
035         initDialogFields(varName + ": %s", modVar.getVarName());
036     }
037
038     public MonitorVariableDialog(java.awt.Frame parent, boolean modal, Monitor
039     edVariable monitoredVar) {
040         super(parent, modal);
041         initComponents();
042         this.modVar = monitoredVar.getModuleVar();
043         initDialogFields(monitoredVar.getPresentation(), modVar.getVarName());
044     }
045
046     private void initDialogFields(String formatText, String varName) {
047         jTextFormat.setText(formatText);
048         jTextFormat.setSelectionStart(0);
049         jTextFormat.setSelectionEnd(formatText.length());
050         jLabelVar.setText(jLabelVar.getText() + varName);
051     }
052
053     public boolean wasApproved() {
054         return wasApproved;
055     }
056
057     public MonitoredVariable getMonitoredVar() {
058         return monitoredVar;
059     }
060
061     /** This method is called from within the constructor to
062      * initialize the form.
063      * WARNING: Do NOT modify this code. The content of this method is
064      * always regenerated by the Form Editor.
065      */
066     @SuppressWarnings("unchecked")
067     // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
068     BEGIN:
069     initComponents
070     {
071         private void initComponents() {
072             // Removed auto-generated code
073         }
074
075         private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { //GE
```



```

N-FIRST:event_jButton1ActionPerformed
176     String presentation = jTextFormat.getText();
177     if (presentation.contains("%s")) {
178         monitoredVar = new MonitoredVariable(modVar, presentation);
179         wasApproved = true;
180         this.setVisible(false);
181     } else {
182         JOptionPane.showMessageDialog(this, "Missing %s in presentation string", "Presentation Error", JOptionPane.ERROR_MESSAGE);
183     }
184 } //GEN-LAST:event_jButton1ActionPerformed
185
186 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) { //GEN-
N-FIRST:event_jButton2ActionPerformed
187     this.setVisible(false);
188 } //GEN-LAST:event_jButton2ActionPerformed
189
190 /**
191  * @param args the command line arguments
192  */
193 public static void main(String args[]) {
194     java.awt.EventQueue.invokeLater(new Runnable() {
195
196         private ModuleVariable modVar = new ModuleVariable("core.version",
197             "d", "202");
198
199         public void run() {
200             MonitorVariableDialog dialog = new MonitorVariableDialog(new javax.swing.JFrame(), true, modVar);
201             dialog.addWindowListener(new java.awt.event.WindowAdapter() {
202                 public void windowClosing(java.awt.event.WindowEvent e) {
203                     System.exit(0);
204                 }
205             });
206             dialog.setVisible(true);
207         }
208     });
209 }
210 // Removed auto-generated code
211 }

```

marg.gui.MainGUIHandler

```
06 package marg.gui;
07
08 import java.util.Collection;
09 import java.util.HashMap;
10 import javax.swing.JToggleButton;
11 import marg.model.MonitoredVariable;
12
13 /**
14  *
15  * @author MARG
16  */
17 public class MainGUIHandler {
18
19     private static MainGUIHandler instance;
20     private ModuleMenu moduleMenu;
21     private MargGUI mainGUI;
22
23     public static MainGUIHandler getInstance() {
24         if (instance == null)
25             instance = new MainGUIHandler();
26         return instance;
27     }
28
29     private MainGUIHandler() {
30     }
31
32     private HashMap<String, RobotModule> moduleMap = new HashMap<String, RobotM
33 odule>();
34     private HashMap<String, JToggleButton> buttonMap = new HashMap<String, JTog
35 gleButton>();
36
37     public void addModuleMapping(RobotModule module) {
38         moduleMap.put(module.getModuleName(), module);
39     }
40
41     public RobotModule getModuleByName(String moduleName) {
42         return moduleMap.get(moduleName);
43     }
44
45     public Collection<RobotModule> getAllModules() {
46         return moduleMap.values();
47     }
48
49     public void addButtonMapping(String moduleName, JToggleButton button) {
50         buttonMap.put(moduleName, button);
51     }
52
53     public JToggleButton getMenuButtonByName(String moduleName) {
54         return buttonMap.get(moduleName);
55     }
56
57     public void setModuleMenu(ModuleMenu modMenu) {
58         this.moduleMenu = modMenu;
59     }
60
61     public MargGUI getMainGUI() {
62         return mainGUI;
63     }
64
65     public ModuleMenu getModuleMenu() {
66         return moduleMenu;
67     }
68 }
```

```
66
67     public void setMainGUI(MargGUI margGUI) {
68         this.mainGUI = margGUI;
69     }
70
71     public void addModule(ModuleTabs moduleTabs) {
72         mainGUI.addModule(moduleTabs);
73     }
74
75     public ModuleTabs getCurrentModule() {
76         return mainGUI.getCurrentModule();
77     }
78
79     public void showModule(String moduleName) {
80         mainGUI.showModule(moduleName);
81     }
82
83     public void addMonitoredVariable(MonitoredVariable monitoredVar) {
84         moduleMenu.addMonitoredVariable(monitoredVar);
85     }
86
87 }
```

marg.gui.MargGUI

```
011 package marg.gui;
012
013 import java.awt.BorderLayout;
014 import java.awt.CardLayout;
015 import java.awt.Component;
016 import java.net.URL;
017 import java.util.List;
018 import java.util.logging.Level;
019 import java.util.logging.Logger;
020 import javax.swing.SwingUtilities;
021 import marg.config.Module;
022 import marg.config.Robot;
023 import marg.config.XMLConfigHandler;
024 import marg.gui.plugin.ModulePlugin;
025 import marg.util.PluginManager;
026 import marg.util.Log;
027
028 /**
029  *
030  * @author MARG
031  */
032 public class MargGUI extends javax.swing.JApplet {
033
034     private static final String ROBOT_CONFIG_FILENAME = "robot.xml";
035     private ModuleMenu menu;
036
037     /** Initializes the applet MargGUI2 */
038     public void init() {
039         try {
040             java.awt.EventQueue.invokeAndWait(new Runnable() {
041                 public void run() {
042                     initComponents();
043                     MainGUIHandler.getInstance().setMainGUI(MargGUI.this);
044
045                     menu = new ModuleMenu(); //MargGUI must be set in MainGUIH
046                     moduleMenuTab.add(menu, BorderLayout.CENTER);
047
048                     Thread loadThread = new Thread(new Runnable() {
049                         public void run() {
050                             try {
051                                 Thread.sleep(300);
052                             } catch (InterruptedException ex) {
053                                 Logger.getLogger(MargGUI.class.getName()).log(
054                                     Level.SEVERE, null, ex);
055                             }
056                             final Robot loadedRobot = loadRobot();
057                             loadTitle(loadedRobot);
058                             loadControlButtons(loadedRobot);
059                             addLoadedModules(loadedRobot.getModules());
060                         }
061                     });
062                     loadThread.start();
063                 }
064             });
065         } catch (Exception ex) {
066             ex.printStackTrace();
067         }
068
069         public void addModule(final ModuleTabs modTab) {
070             SwingUtilities.invokeLater(new Runnable() {
```

```

071
072         public void run() {
073             moduleTabs.add(modTab, modTab.getModuleName());
074         }
075     });
076 }
077
078 public void showModule(final String moduleName) {
079     SwingUtilities.invokeLater(new Runnable() {
080
081         public void run() {
082             CardLayout cl = (CardLayout) moduleTabs.getLayout();
083             cl.show(moduleTabs, moduleName);
084         }
085     });
086 }
087
088 private Robot loadRobot() {
089     XMLConfigHandler configHandler = new XMLConfigHandler();
090     java.net.URL codebaseURL = MargGUI.this.getCodeBase();
091     System.out.println("CodeBase: " + codebaseURL);
092     if (XMLConfigHandler.isLocalURL(codebaseURL)) {
093         codebaseURL = XMLConfigHandler.fixLocalUrl(codebaseURL);
094     }
095     URL configFile = XMLConfigHandler.getFileURL(codebaseURL, ROBOT_CONFIG
096 _FILENAME);
097     return configHandler.getConfigFrom(configFile);
098 }
099
100 private void loadTitle(Robot loadedRobot) {
101     menu.setRobotName(loadedRobot.getRobotName());
102 }
103
104 private void loadControlButtons(Robot robot) {
105     menu.setStartButton(robot.getStartButton());
106     menu.setPauseButton(robot.getPauseButton());
107     menu.setStopButton(robot.getStopButton());
108 }
109
110 //TODO: Refactor this out somewhere else? Keep margGUI as clean as possibl
111 e!
112 private void addLoadedModules(List<Module> loadedModules) {
113     for (Module mod : loadedModules) {
114         //Adding a module
115         ModuleTabs module = new ModuleTabs(mod.getModuleName(), mod.getMod
116 uleHost(), mod.getPort(), mod.isAutoConnect());
117         List<Class> availablePlugins = PluginManager.getInstance().getAvai
118 lableModulePlugins(); //Get available plugins
119         for (String pluginName : mod.getModulePlugins()) {
120             boolean foundAvailablePlugin = false;
121             for (Class aClass : availablePlugins) {
122                 if (aClass.getSimpleName().equalsIgnoreCase(pluginName)) {
123                     //Is the requested plugin available?
124                     Object instance = PluginManager.getInstance().getInsta
125 nceOfClass(aClass); //Create an instance if so
126                     if (instance != null) {
127                         module.addModulePlugin((ModulePlugin) instance); /
128 /Add it to the module that requested the plugin
129                         foundAvailablePlugin = true;
130                         Log.GlobalLogger.info("Added plugin '" + pluginNam
131 e + "' for module " + mod.getModuleName());
132                     }
133                     break; //Get out of the for-
134 loop, we already found the plugin we needed.

```

```

126         }
127     }
128     if (!foundAvailablePlugin) {
129         Log.GlobalLogger.info("Could not load requested plugin '"
+ pluginName + "' for module " + mod.getModuleName());
130     }
131 }
132 //Adding it to the gui
133 menu.addModule(module);
134 }
135 }
136
137 /**
138  * Gets currently visible module in the cardlayout
139  * @return The visible ModuleTabs object. null if no object is visisble.
140  */
141 public ModuleTabs getCurrentModule() {
142     Component[] comp = moduleTabs.getComponents();
143     for (Component com : comp) {
144         if (com.isVisible() && com instanceof ModuleTabs) {
145             return (ModuleTabs) com;
146         }
147     }
148     return null;
149 }
150
151 /** This method is called from within the init() method to
152  * initialize the form.
153  * WARNING: Do NOT modify this code. The content of this method is
154  * always regenerated by the Form Editor.
155  */
156 @SuppressWarnings("unchecked")
157 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN: initComponents
158 private void initComponents() {
201 } // Removed auto-generated code
202 // Variables declaration - do not modify//GEN-BEGIN:variables
203 private javax.swing.JPanel jPanel1;
204 private javax.swing.JPanel moduleMenuTab;
205 private javax.swing.JPanel moduleTabs;
206 // End of variables declaration//GEN-END:variables
207
208 /**
209  * Called by the browser when an applet is closed.. Cleanup!
210  */
211 @Override
212 public void destroy() {
213     Component[] comps = moduleTabs.getComponents();
214     for (Component comp : comps) {
215         if (comp instanceof RobotModule) {
216             RobotModule module = (RobotModule) comp;
217             Log.GlobalLogger.info("Closing Module: " + module.getModuleNam
e());
218             module.getXMLClientHandler().disconnect();
219             module.stopPlugins();
220         }
221     }
222 }
223 }

```

marg.gui.ModuleMenu

```
006 package marg.gui;
007
008 import java.awt.Color;
009 import java.awt.Dimension;
010 import java.awt.GridBagConstraints;
011 import java.awt.event.ActionEvent;
012 import java.awt.event.ActionListener;
013 import java.util.Collection;
014 import java.util.List;
015 import java.util.logging.Level;
016 import java.util.logging.Logger;
017 import javax.swing.ButtonGroup;
018 import javax.swing.DefaultListModel;
019 import javax.swing.ImageIcon;
020 import javax.swing.JOptionPane;
021 import javax.swing.JToggleButton;
022 import javax.swing.SwingConstants;
023 import javax.swing.SwingUtilities;
024 import marg.config.ControlButton;
025 import marg.gui.dialogs.AddModuleDialog;
026 import marg.gui.dialogs.MonitorVariableDialog;
027 import marg.gui.plugin.ModulePlugin;
028 import marg.images.ResUtils;
029 import marg.model.MonitoredVariable;
030 import marg.util.Log;
031 import marg.util.PluginManager;
032
033 /**
034  *
035  * @author MARG
036  */
037 public class ModuleMenu extends javax.swing.JPanel implements Runnable {
038
039     private static final long BUTTON_STATUS_UPDATE_DELAY = 500;
040     private int moduleRows = 0;
041     private ControlButton startButton;
042     private ControlButton pauseButton;
043     private ControlButton stopButton;
044     private ButtonGroup moduleBtnGroup = new ButtonGroup();
045     private ImageIcon greenIcon = ResUtils.getImageIcon("GreenButton20px.gif");
046     private ImageIcon yellowIcon = ResUtils.getImageIcon("YellowButton20px.gif");
047
048     ;
049     private ImageIcon redIcon = ResUtils.getImageIcon("RedButton20px.gif");
050
051     ;
052     private boolean buttonStatusUpdateThreadRunning = false;
053
054     /** Creates new form Modules */
055     public ModuleMenu() {
056         initComponents();
057         MainGUIHandler.getInstance().setModuleMenu(this);
058         jListMonitored.setModel(new DefaultListModel());
059         startUpdateThread();
060     }
061
062     public void addModule(RobotModule module) {
063         String buttonName = module.getModuleName();
064         MainGUIHandler handler = MainGUIHandler.getInstance();
065         handler.addModuleMapping(module);
```

```

066         handler.addModule((ModuleTabs) module);
067         this.addButton(buttonName);
068         handler.showModule(module.getModuleName());
069     }
070
071     public void addMonitoredVariable(MonitoredVariable monitoredVar) {
072         DefaultListModel model = (DefaultListModel) jListMonitored.getModel();
073         model.addElement(monitoredVar);
074     }
075
076     /** This method is called from within the constructor to
077      * initialize the form.
078      * WARNING: Do NOT modify this code. The content of this method is
079      * always regenerated by the Form Editor.
080      */
081     @SuppressWarnings("unchecked")
082     // <editor-fold defaultstate="collapsed" desc="Generated Code"> // GEN-
BEGIN: initComponents
083     private void initComponents() {
084     } // Removed auto-generated code
085
086     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { // GE
N-FIRST:event_jButton1ActionPerformed
087         AddModuleDialog dialog = new AddModuleDialog(null, true);
088         dialog.setLocation(200, 200); // TODO: Un-hardcode coordinates?
089         dialog.setVisible(true);
090         dialog.dispose();
091         if (dialog.wasApproved()) {
092             String moduleName = dialog.getModuleName();
093             String host = dialog.getHost();
094             int port = dialog.getPort();
095             boolean autoConnect = dialog.getAutoConnect();
096             if (!moduleName.equals("")) {
097                 RobotModule module = new ModuleTabs(moduleName, host, port, au
toConnect);
098                 this.addModule(module);
099             } else {
100                 JOptionPane.showMessageDialog(this, "Cannot create a module wi
thout a name");
101             }
102         } // GEN-LAST:event_jButton1ActionPerformed
103
104     private void jBtnAddPluginActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST:event_jBtnAddPluginActionPerformed
105         List<Class> availablePlugins = PluginManager.getInstance().getAvailabl
eModulePlugins();
106         String[] pluginNames = new String[availablePlugins.size()];
107         for (int i = 0; i < availablePlugins.size(); i++) {
108             pluginNames[i] = availablePlugins.get(i).getSimpleName();
109         }
110         String selectedName = (String) JOptionPane.showInputDialog(null,
111             "Choose Plugin", "Add Plugin to active module", JOptionPane.QU
ESTION_MESSAGE,
112             null, pluginNames, null);
113         if (selectedName != null) { // Was a plugin chosen?
114             for (Class aClass : availablePlugins) { // Find the Class for the c
hosen plugin
115                 if (aClass.getSimpleName().equals(selectedName)) {
116                     ModuleTabs module = MainGUIHandler.getInstance().getCurren
tModule();
117                     ModulePlugin plugin = (ModulePlugin) PluginManager.getInst
ance().getInstanceOfClass(aClass);
118                     if (module != null && plugin != null) {

```



```

359             module.addModulePlugin(plugin);
360         }
361     }
362 }
363 }
364
365 }//GEN-LAST:event_jBtnAddPluginActionPerformed
366
367 private void stopBTNActionPerformed(java.awt.event.ActionEvent evt) { //GEN
- FIRST:event_stopBTNActionPerformed
368     if (stopButton != null) {
369         RobotModule module = MainGUIHandler.getInstance().getModuleByName(
stopButton.getModuleName());
370         if (module != null) {
371             sendCmdThroughModule(module, stopButton.getCommand());
372         }
373     }
374 }//GEN-LAST:event_stopBTNActionPerformed
375
376 private void pauseBTNActionPerformed(java.awt.event.ActionEvent evt) { //GE
N-FIRST:event_pauseBTNActionPerformed
377     if (pauseButton != null) {
378         RobotModule module = MainGUIHandler.getInstance().getModuleByName(
pauseButton.getModuleName());
379         if (module != null) {
380             sendCmdThroughModule(module, pauseButton.getCommand());
381         }
382     }
383 }//GEN-LAST:event_pauseBTNActionPerformed
384
385 private void startBTNActionPerformed(java.awt.event.ActionEvent evt) { //GE
N-FIRST:event_startBTNActionPerformed
386     if (startButton != null) {
387         RobotModule module = MainGUIHandler.getInstance().getModuleByName(
startButton.getModuleName());
388         if (module != null) {
389             sendCmdThroughModule(module, startButton.getCommand());
390         } else {
391             Log.GlobalLogger.info("No Module (" + startButton.getModuleNam
e() + ") to send start cmd (" + startButton.getCommand() + ") to");
392         }
393     } else {
394         Log.GlobalLogger.fine("Start Button not configured");
395     }
396 }//GEN-LAST:event_startBTNActionPerformed
397
398 private void jListMonitoredMouseClicked(java.awt.event.MouseEvent evt) { //
GEN-FIRST:event_jListMonitoredMouseClicked
399     if (SwingUtilities.isRightMouseButton(evt)) {
400         jListMonitored.setSelectedIndex(jListMonitored.locationToIndex(evt
.getPoint()));
401         Object selectedObj = jListMonitored.getSelectedValue();
402         if (selectedObj != null) {
403             jPopupMonitorList.show(jListMonitored, evt.getX(), evt.getY())
;
404         }
405     }
406 }//GEN-LAST:event_jListMonitoredMouseClicked
407
408 private void jMenuEditPresentationActionPerformed(java.awt.event.ActionEve
nt evt) { //GEN-FIRST:event_jMenuEditPresentationActionPerformed
409     Object selectedObj = jListMonitored.getSelectedValue();
410     if (selectedObj != null) {
411         MonitoredVariable monitoredVar = (MonitoredVariable) selectedObj;

```

```

412         MonitorVariableDialog dialog = new MonitorVariableDialog(null, true, monitoredVar);
413         dialog.setLocation(200, 200);
414         dialog.setVisible(true);
415         if (dialog.wasApproved()) {
416             MonitoredVariable newMonVar = dialog.getMonitoredVar();
417             monitoredVar.setPresentation(newMonVar.getPresentation());
418         }
419         jListMonitored.clearSelection();
420     }
421 }
422 //GEN-LAST:event_jMenuEditPresentationActionPerformed
423
424 private void sendCmdThroughModule(RobotModule module, String cmd) {
425     module.getXMLClientHandler().sendCmd(cmd);
426 }
427 // Variables declaration - do not modify//GEN-BEGIN:variables
428 private javax.swing.JButton jBtnAddPlugin;
429 private javax.swing.JButton jButton1;
430 private javax.swing.JLabel jLabel1;
431 private javax.swing.JLabel jLabel2;
432 private javax.swing.JLabel jLabel3;
433 private javax.swing.JLabel jLabel4;
434 private javax.swing.JList jListMonitored;
435 private javax.swing.JMenuItem jMenuItemEditPresentation;
436 private javax.swing.JPanel jModuleButtons;
437 private javax.swing.JPanel jPanel1;
438 private javax.swing.JPanel jPanel2;
439 private javax.swing.JPanel jPanel3;
440 private javax.swing.JPanel jPanel4;
441 private javax.swing.JPanel jPanel5;
442 private javax.swing.JPopupMenu jPopupMenuMonitorList;
443 private javax.swing.JLabel jRobotName;
444 private javax.swing.JScrollPane jScrollPane1;
445 private javax.swing.JScrollPane jScrollPane2;
446 private javax.swing.JButton pauseBTN;
447 private javax.swing.JButton startBTN;
448 private javax.swing.JButton stopBTN;
449 // End of variables declaration//GEN-END:variables
450
451 private synchronized void addButton(String buttonName) {
452     marg.util.Log.GlobalLogger.info("Adding a button for: " + buttonName);
453     final JToggleButton button = new JToggleButton(buttonName, redIcon);
454     MainGUIHandler.getInstance().addButtonMapping(buttonName, button);
455     button.setHorizontalAlignment(SwingConstants.LEFT);
456     button.setPreferredSize(new Dimension(170, 30));
457     button.addActionListener(new ActionListener() {
458         public void actionPerformed(ActionEvent e) {
459             String moduleName = button.getText();
460             MainGUIHandler.getInstance().showModule(moduleName);
461         }
462     });
463     GridBagConstraints c = new GridBagConstraints();
464     c.gridx = 1;
465     c.gridy = ++moduleRows;
466     jModuleButtons.add(button, c);
467     moduleBtnGroup.add(button);
468     moduleBtnGroup.setSelected(button.getModel(), true);
469 }
470
471 public void setPauseButton(ControlButton pauseButton) {
472     this.pauseButton = pauseButton;
473 }
474

```

```

475     public void setStartButton(ControlButton startButton) {
476         this.startButton = startButton;
477     }
478
479     public void setStopButton(ControlButton stopButton) {
480         this.stopButton = stopButton;
481     }
482
483     public void setRobotName(String robotName) {
484         jRobotName.setText(robotName);
485     }
486
487     private void startUpdateThread() {
488         buttonStatusUpdateThreadRunning = true;
489         Thread t = new Thread(this);
490         t.start();
491     }
492
493     public void run() {
494         while (buttonStatusUpdateThreadRunning) {
495             Collection<RobotModule> modules = MainGUIHandler.getInstance().get
AllModules();
496             for (RobotModule module : modules) {
497                 String moduleName = module.getModuleName();
498                 Color statusColor = module.getModuleStatusColor();
499                 JToggleButton moduleButton = MainGUIHandler.getInstance().getM
enuButtonByName(moduleName);
500                 if (moduleButton != null) {
501                     if (statusColor == Color.GREEN) {
502                         moduleButton.setIcon(greenIcon);
503                     } else if (statusColor == Color.YELLOW) {
504                         moduleButton.setIcon(yellowIcon);
505                     } else if (statusColor == Color.RED) {
506                         moduleButton.setIcon(redIcon);
507                     }
508                 }
509             }
510             updateMonitoredList();
511             try {
512                 Thread.sleep(BUTTON_STATUS_UPDATE_DELAY);
513             } catch (InterruptedException ex) {
514                 Logger.getLogger(ModuleMenu.class.getName()).log(Level.SEVERE,
null, ex);
515             }
516         }
517     }
518
519     private void updateMonitoredList() {
520         SwingUtilities.invokeLater(new Runnable() {
521             public void run() {
522                 jListMonitored.repaint();
523             }
524         });
525     }
526 }

```

marg.gui.ModuleTabs

```
011 package marg.gui;
012
013 import java.awt.Color;
014 import marg.gui.plugin.ModulePlugin;
015 import java.beans.PropertyChangeEvent;
016 import java.beans.PropertyChangeListener;
017 import java.util.ArrayList;
018 import java.util.logging.Level;
019 import java.util.logging.Logger;
020 import javax.swing.DefaultComboBoxModel;
021 import javax.swing.DefaultListModel;
022 import javax.swing.JFrame;
023 import javax.swing.JOptionPane;
024 import javax.swing.JPanel;
025 import javax.swing.SwingUtilities;
026 import javax.swing.text.BadLocationException;
027 import javax.swing.tree.DefaultMutableTreeNode;
028 import javax.swing.tree.TreePath;
029 import marg.gui.dialogs.AddStatusLimitDialog;
030 import marg.gui.dialogs.MonitorVariableDialog;
031 import marg.handlers.XMLClientHandler;
032 import marg.model.ModuleVariable;
033 import marg.model.MonitoredVariable;
034 import marg.model.StatusVariable;
035 import marg.model.ValueLimit;
036 import marg.model.XMLClient;
037 import marg.model.plugin.RawDataPlugin;
038 import marg.model.plugin.VarDataPlugin;
039 import marg.util.Log;
040
041 /**
042  *
043  * @author MARG
044  */
045 public class ModuleTabs extends javax.swing.JPanel implements RobotModule, PropertyChangeListener {
046
047     private static final long PLUGIN_UPDATE_INTERVAL = 1000;
048     private ArrayList<ModulePlugin> modPlugins = new ArrayList<ModulePlugin>();
049
050     private boolean pluginUpdateThreadRunning = false;
051     private boolean hasSubscribed = false;
052     XMLClientHandler handler;
053     VariableTree tree;
054
055     /** Creates new form ModuleTabs */
056     public ModuleTabs(String name) {
057         this(name, "", 0, false);
058     }
059
060     public ModuleTabs(String name, String host, int port, boolean autoConnect)
061     {
062         initComponents();
063         jList1.setModel(new DefaultListModel());
064         this.setName(name);
065         hostTF.setText(host);
066         if (port != 0) {
067             portTF.setText(String.valueOf(port));
068         }
069         initXMLClient();
070         startPluginUpdateThread();
071         if (autoConnect) {
```

```

070         connect();
071     }
072 }
073
074 private void initXMLClient() {
075     handler = new XMLClientHandler(new XMLClient());
076     handler.getXMLParser().addParsePlugin(new RawDataPlugin()); //TODO: Sh
ould this be done manually here?
077     handler.getXMLParser().addParsePlugin(new VarDataPlugin());
078 }
079
080 public void addModulePlugin(ModulePlugin modPlugin) {
081     modPlugins.add(modPlugin);
082     JPanel pluginPanel = modPlugin.getJPanel();
083     pluginPanel.setName(modPlugin.getPluginName());
084     tabsPN.add(pluginPanel);
085     modPlugin.setXMLClientHandler(handler); //Skal have handler fÅr den k
an starte
086     modPlugin.startPlugin(); //Plugin startes og kan bruge XMLClientHandle
r osv.
087 }
088
089 /** This method is called from within the constructor to
090  * initialize the form.
091  * WARNING: Do NOT modify this code. The content of this method is
092  * always regenerated by the Form Editor.
093  */
094 @SuppressWarnings("unchecked")
095 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
096 private void initComponents() {
522     }// Removed auto-generated code
523
524 private void jConnectBtnActionPerformed(java.awt.event.ActionEvent evt) {/
//GEN-FIRST:event_jConnectBtnActionPerformed
525     connect();
526 }//GEN-LAST:event_jConnectBtnActionPerformed
527
528 private void cmdBTActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cmdBTActionPerformed
529     String cmd = (String) jCmdComboBox.getSelectedItem();
530     System.out.println("Cmd: " + cmd);
531
532     if (!cmd.equals("") && !cmd.equals(" ")) {
533         //Send Cmd
534         handler.sendCmd(cmd);
535         jTextArea2.append("\n-> " + cmd);
536
537         //Add to combo-box if needed
538         DefaultComboBoxModel model = (DefaultComboBoxModel) jCmdComboBox.g
etModel();
539         int index = model.indexOf(cmd);
540         if (index == -1) { //new command
541             model.insertElementAt(cmd, 0);
542         }
543         jCmdComboBox.setSelectedItem("");
544     }
545 }//GEN-LAST:event_cmdBTActionPerformed
546
547 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GE
N-FIRST:event_jButton2ActionPerformed
548     handler.sendCmd("var allcopy");
549 }//GEN-LAST:event_jButton2ActionPerformed
550

```

```

551     private void jTree1MouseClicked(java.awt.event.MouseEvent evt) { //GEN-
FIRST:event_jTree1MouseClicked
552         if (SwingUtilities.isRightMouseButton(evt)) {
553             TreePath rightClickedPath = jTree1.getClosestPathForLocation(evt.g
etX(), evt.getY());
554             if (rightClickedPath != null) {
555                 jTree1.setSelectionPath(rightClickedPath);
556                 DefaultMutableTreeNode node = (DefaultMutableTreeNode) jTree1.
getSelectionPath().getLastPathComponent();
557                 if (node.isLeaf() && node.getUserObject() instanceof ModuleVar
iable) {
558                     ModuleVariable modVar = (ModuleVariable) node.getUserObjec
t();
559                     if (statusVariableExists(modVar)) {
560                         jMenuAddToStatusTab.setVisible(false);
561                     } else {
562                         jMenuAddToStatusTab.setVisible(true);
563                     }
564                     jPopupMenuVariables.show(jTree1, evt.getX(), evt.getY());
565                     Log.GlobalLogger.fine("VarTree: Right-
Clicked on " + modVar);
566                 }
567             }
568         }
569     } //GEN-LAST:event_jTree1MouseClicked
570
571     private void jMenuChangeValueActionPerformed(java.awt.event.ActionEvent ev
t) { //GEN-FIRST:event_jMenuChangeValueActionPerformed
572         DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) jTree1.
getSelectionPath().getLastPathComponent();
573         if (selectedNode.getUserObject() instanceof ModuleVariable) {
574             modifyModuleVariable((ModuleVariable) selectedNode.getUserObject()
);
575         }
576     } //GEN-LAST:event_jMenuChangeValueActionPerformed
577
578     private void jCmdComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
//GEN-FIRST:event_jCmdComboBoxActionPerformed
579         if (evt.getActionCommand().equals("comboBoxEdited")) {
580             cmdBTActionPerformed(null); //Pushes send
581         }
582     } //GEN-LAST:event_jCmdComboBoxActionPerformed
583
584     private void jMenuAddToStatusTabActionPerformed(java.awt.event.ActionEvent
evt) { //GEN-FIRST:event_jMenuAddToStatusTabActionPerformed
585         DefaultListModel model = (DefaultListModel) jList1.getModel();
586         DefaultMutableTreeNode node = (DefaultMutableTreeNode) jTree1.getSelec
tionPath().getLastPathComponent();
587         if (node.isLeaf() && node.getUserObject() instanceof ModuleVariable) {
588             ModuleVariable modVar = (ModuleVariable) node.getUserObject();
589             StatusVariable statusVar = new StatusVariable(modVar);
590             model.addElement(statusVar);
591         }
592     } //GEN-LAST:event_jMenuAddToStatusTabActionPerformed
593
594     private void jList1MouseClicked(java.awt.event.MouseEvent evt) { //GEN-
FIRST:event_jList1MouseClicked
595         if (SwingUtilities.isRightMouseButton(evt)) {
596             Object obj = jList1.getSelectedValue();
597             if (obj != null) {
598                 jPopupMenuStatus.show(jList1, evt.getX(), evt.getY());
599             }
600         }
601     } //GEN-LAST:event_jList1MouseClicked

```

```

602
603     private void jMenuItemAddLimitActionPerformed(java.awt.event.ActionEvent evt)
604     { //GEN-FIRST:event_jMenuItemAddLimitActionPerformed
605         Object obj = jList1.getSelectedValue();
606         if (obj != null) {
607             StatusVariable statVar = (StatusVariable) obj;
608             AddStatusLimitDialog dialog = new AddStatusLimitDialog(null, true,
609                 statVar);
610             dialog.setLocation(250, 200); //TODO: Un-
611             //hardcode location? Works for now.
612             dialog.setTitle("Add Value Limit");
613             dialog.setVisible(true);
614             if (dialog.wasApproved()) {
615                 ValueLimit limit = dialog.getValueLimit();
616                 statVar.setLimit(limit);
617             }
618         }
619     } //GEN-LAST:event_jMenuItemAddLimitActionPerformed
620
621     private void jMenuItemMonitorVariableActionPerformed(java.awt.event.ActionEven
622     t evt) { //GEN-FIRST:event_jMenuItemMonitorVariableActionPerformed
623         DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) jTree1.
624         getSelectionPath().getLastPathComponent();
625         if (selectedNode.getUserObject() instanceof ModuleVariable) {
626             MonitorVariableDialog dialog = new MonitorVariableDialog(null, tru
627             e, (ModuleVariable) selectedNode.getUserObject());
628             dialog.setLocation(200, 200);
629             dialog.setVisible(true);
630             if (dialog.wasApproved()) {
631                 MonitoredVariable monitoredVar = dialog.getMonitoredVar();
632                 if (monitoredVar != null) {
633                     MainGUIHandler.getInstance().addMonitoredVariable(monitore
634                     dVar);
635                 }
636             }
637         }
638     } //GEN-LAST:event_jMenuItemMonitorVariableActionPerformed
639
640     private boolean statusVariableExists(ModuleVariable modVar) {
641         DefaultListModel model = (DefaultListModel) jList1.getModel();
642         for (Object obj : model.toArray()) {
643             StatusVariable statVar = (StatusVariable) obj;
644             if (statVar.getModuleVar().getVarName().equals(modVar.getVarName()
645             )) {
646                 return true;
647             }
648         }
649         return false;
650     }
651
652     private boolean isStatusVariablesCorrect() {
653         boolean isCorrect = true;
654         DefaultListModel model = (DefaultListModel) jList1.getModel();
655         for (Object obj : model.toArray()) {
656             StatusVariable statVar = (StatusVariable) obj;
657             if (!statVar.isStatusCorrect()) {
658                 isCorrect = false;
659                 break;
660             }
661         }
662         return isCorrect;
663     }
664
665     // Variables declaration - do not modify //GEN-BEGIN:variables
666     // Removed auto-generated code

```

```

697
698     public void propertyChange(PropertyChangeEvent evt) {
699         if (evt.getPropertyName().equals(RawDataPlugin.SUBSCRIBE_RAWDATA)) {
700             newRawData(evt);
701         } else if (evt.getPropertyName().equals(VarDataPlugin.SUBSCRIBE_VARDAT
A)) {
702             newVarData(evt);
703         }
704     }
705
706     private void modifyModuleVariable(ModuleVariable modVar) {
707         String newValue = (String) JOptionPane.showInputDialog(this, "Please e
nter the new value:", modVar.getVarName(), JOptionPane.QUESTION_MESSAGE, null, nul
l, "" + modVar.getValue());
708         if (newValue != null && !newValue.equals("")) {
709             handler.setVariable(modVar.getVarName(), newValue);
710         }
711     }
712
713     public static void main(String[] args) {
714         JFrame frame = new JFrame("Title");
715         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
716         frame.setSize(800, 600);
717         ModuleTabs modTabs = new ModuleTabs("RHD Test");
718         frame.getContentPane().add(modTabs);
719         frame.setVisible(true);
720     }
721
722     public Color getModuleStatusColor() {
723         boolean isConnected = handler.isConnected();
724         if (isConnected) {
725             if (isStatusVariablesCorrect()) {
726                 return Color.GREEN;
727             } else {
728                 return Color.YELLOW;
729             }
730         } else {
731             return Color.RED;
732         }
733     }
734
735     public String getModuleName() {
736         return this.getName();
737     }
738
739     private void newRawData(PropertyChangeEvent evt) {
740         String newRawData = (String) evt.getNewValue();
741         if (jTextArea2.getLineCount() > 1000) {
742             try {
743                 int cutOffset = jTextArea2.getLineStartOffset(500);
744                 int endOffset = jTextArea2.getLineEndOffset(jTextArea2.getLine
Count() - 1);
745                 jTextArea2.setText(jTextArea2.getText(cutOffset, endOffset -
cutOffset));
746             } catch (BadLocationException ex) {
747                 Log.GlobalLogger.info("RawData, Bad Location offset: " + ex.of
fsetRequested());
748             }
749         }
750         jTextArea2.append(newRawData);
751         try {
752             int offset = jTextArea2.getLineStartOffset(jTextArea2.getLineCount
() - 1);
753             jTextArea2.setCaretPosition(offset);

```



```

754         } catch (BadLocationException ex) {
755             Logger.getLogger(ModuleTabs.class.getName()).log(Level.SEVERE, nul
l, ex);
756         }
757     }
758
759     private void newVarData(PropertyChangeEvent evt) {
760         ModuleVariable var = (ModuleVariable) evt.getNewValue();
761         tree.updateVarTreeStructure(var);
762         tree.updateVarTreeView();
763         this.updateStatusListView();
764     }
765
766     private void updateStatusListView() {
767         jList1.repaint();
768     }
769
770     public XMLClientHandler getXMLClientHandler() {
771         return handler;
772     }
773
774     private void connect() {
775         boolean isConnected = handler.connect(hostTF.getText(), Integer.parseI
nt(portTF.getText()));
776         if (isConnected) {
777             handler.sendCmd("var allcopy");
778         }
779         if (!hasSubscribed) {
780             handler.addPropertyChangeListener(this, RawDataPlugin.SUBSCRIBE_
RAWDATA, VarDataPlugin.SUBSCRIBE_VARDATA);
781             tree = new VariableTree(jTree1, this.getName());
782             hasSubscribed = true;
783         }
784     }
785
786     public void removeModulePlugin(ModulePlugin modPlugin) {
787         tabsPN.remove(modPlugin.getJPanel());
788         modPlugins.remove(modPlugin);
789         modPlugin.stopPlugin();
790     }
791
792     public void stopPlugins() {
793         for (ModulePlugin plugin : modPlugins) {
794             plugin.stopPlugin();
795             tabsPN.remove(plugin.getJPanel());
796         }
797     }
798
799     private void startPluginUpdateThread() {
800         pluginUpdateThreadRunning = true;
801         Thread t = new Thread(new Runnable() {
802             public void run() {
803                 while (pluginUpdateThreadRunning) {
804                     for (ModulePlugin plugin : modPlugins) {
805                         plugin.doUpdate();
806                     }
807                     try {
808                         Thread.sleep(PLUGIN_UPDATE_INTERVAL);
809                     } catch (InterruptedException ex) {
810                         Logger.getLogger(ModuleTabs.class.getName()).log(Level
.SEVERE, null, ex);
811                     }
812                 }
813             }

```

```

814         });
815         t.start();
816     }
817 }

```

marg.gui.RobotModule

```

06 package marg.gui;
07
08 import java.awt.Color;
09 import marg.gui.plugin.ModulePlugin;
10 import marg.handlers.XMLClientHandler;
11
12 /**
13  *
14  * @author MARG
15  */
16 public interface RobotModule {
17
18     /**
19      * Adds a plugin to the module
20      * @param modPlugin the plugin to be added
21      */
22     public void addModulePlugin(ModulePlugin modPlugin);
23
24     /**
25      * Removes a plugin from the module
26      * @param modPlugin the plugin to be removed
27      */
28     public void removeModulePlugin(ModulePlugin modPlugin);
29
30     /**
31      * Gets the name of this module. Used on its button in the menu
32      * @return the name of this module
33      */
34     public String getModuleName();
35
36     /**
37      * Gets the current status of the module as a Color object
38      * @return a Color object representing the module's status
39      */
40     public Color getModuleStatusColor();
41
42     /**
43      * Get's the underlying XMLClientHandler used by the module.
44      * The handler is the one used by the module to communicate
45      * with an actual module on the robot.
46      * @return the underlying handler used by the module
47      */
48     public XMLClientHandler getXMLClientHandler();
49
50     /**
51      * Stops all plugins running on the module. Mainly used to do cleanup.
52      */
53     public void stopPlugins();
54
55 }

```

marg.gui.VariableTree

```
06 package marg.gui;
07
08 import javax.swing.JTree;
09 import javax.swing.SwingUtilities;
10 import javax.swing.tree.DefaultMutableTreeNode;
11 import javax.swing.tree.DefaultTreeModel;
12 import marg.model.ModuleVariable;
13
14 /**
15  *
16  * @author MARG
17  */
18 public class VariableTree {
19
20     private DefaultMutableTreeNode varTop;
21     private JTree jTree;
22     private String moduleName;
23
24     public VariableTree(JTree jTree, String moduleName) {
25         this.jTree = jTree;
26         this.moduleName = moduleName;
27     }
28
29     public void updateVarTreeView() {
30         SwingUtilities.invokeLater(new Runnable() {
31             public void run() {
32                 jTree.updateUI();
33             }
34         });
35     }
36
37     public void updateVarTreeStructure(ModuleVariable var) {
38         String shortName = "";
39         boolean foundExistingLeaf = false;
40         boolean isOnLeafNode = false;
41         DefaultMutableTreeNode node;
42         if (varTop == null) {
43             varTop = new DefaultMutableTreeNode(moduleName); //Top node represents module
44             jTree.setModel(new DefaultTreeModel(varTop));
45         }
46         if (var.getVarName() != null && var.getValue() != null && !var.getVarName().equals("")) {
47             node = (DefaultMutableTreeNode) jTree.getModel().getRoot();
48             String[] varStringTree = var.getVarName().split("\\.");
49             shortName = varStringTree[varStringTree.length - 1];
50
51             //Loop finds the node to attach the new leafnode (or updates existing leaf node)
52             for (int i = 0; i < varStringTree.length; i++) { //goes through each part of struct.struct.struct.leaf sequentially
53                 String sNode = varStringTree[i];
54                 if (i == varStringTree.length - 1) {
55                     isOnLeafNode = true;
56                 }
57                 DefaultMutableTreeNode oldNode = node;
58                 for (int j = 0; j < node.getChildCount(); j++) {
59                     DefaultMutableTreeNode aNode = (DefaultMutableTreeNode) node.getChildAt(j);
60                     Object userObject = aNode.getUserObject();
61                     if (userObject instanceof String) {
62                         //Structure node, let's compare to our current sNode
```

```

63         if (((String) userObject).equals(sNode)) {
64             node = aNode;
65             break;
66         }
67     } else if (userObject instanceof ModuleVariable) {
68         if (((ModuleVariable) userObject).getShortVarName().equals(
69             shortName) && isOnLeafNode) {
70             foundExistingLeaf = true;
71             node = aNode;
72             break;
73         }
74     }
75     if (oldNode == node && !isOnLeafNode) { //If no matches and not
76         a leaf we make a struct node
77         DefaultMutableTreeNode newNode = new DefaultMutableTreeNode
78         (sNode);
79         node.add(newNode);
80         node = newNode;
81     }
82     if (foundExistingLeaf) {
83         ModuleVariable moduleVar = (ModuleVariable) node.getUserObject(
84         );
85         moduleVar.setValue(var.getValue());
86     } else {
87         DefaultMutableTreeNode leafNode = new DefaultMutableTreeNode(var);
88         //Var has a toString that provides proper formatting
89         node.add(leafNode);
90     }
91 }

```

marg.gui.plugin.ModulePlugin

```
06 package marg.gui.plugin;
07
08 import javax.swing.JPanel;
09 import marg.handlers.XMLClientHandler;
10
11 /**
12  * @author MARG
13  */
14
15 public interface ModulePlugin {
16
17     /**
18      * Offers the handler to your plugin.
19      * The handler is responsible for communicating with the parent module and
20      * is also where you should register for propertyChangeEvents and add new X
21      * MLParsePlugins.
22      * This method will by contract always be called before startPlugin();
23      * @param handler the XMLClientHandler object sent from the Module
24      */
25     public void setXMLClientHandler(XMLClientHandler handler);
26
27     /**
28      * Gets the name of your plugin.
29      * Used as the header for the tab of your plugin.
30      * @return a String representing the name of your plugin
31      */
32     public String getPluginName();
33
34     /**
35      * Return the visual representation of your plugin here.
36      * Parent module calls this method to retrieve a JPanel and adds it to its
37      * tabbed pane.
38      * @return a JPanel object created by your plugin
39      */
40     public JPanel getJPanel();
41
42     /**
43      * Called by the parent module to start a plugin.
44      * Use this method to initialize ressources that are not needed until
45      * your plugin is actually started
46      */
47     public void startPlugin();
48
49     /**
50      * Called by the parent module during cleanup process when the module or en
51      * tire applet is closed.
52      * Clean-up of ressources should be placed here.
53      */
54     public void stopPlugin();
55
56     /**
57      * Called by parent module regularly every second by default.
58      * Saves you having to make a Thread if you need a simple regular update in
59      * your plugin.
60      */
61     public void doUpdate();
62 }
```

marg.gui.plugin.SMRPlugin

```
011 package marg.gui.plugin;
012
013 import marg.gui.plugin.smr.*;
014 import java.awt.BorderLayout;
015 import java.beans.PropertyChangeEvent;
016 import java.beans.PropertyChangeListener;
017 import javax.swing.JFrame;
018 import javax.swing.JPanel;
019 import marg.handlers.XMLClientHandler;
020 import marg.model.ModuleVariable;
021 import marg.model.plugin.VarDataPlugin;
022
023 /**
024  *
025  * @author MARG
026  */
027 public class SMRPlugin extends javax.swing.JPanel implements ModulePlugin, PropertyChangeListener {
028
029     public static final int ENCODER_UPDATE_DELAY = 1000;
030     private static final String LINESENSOR_VARNAME = "rhd.linesensor";
031     private static final String IRSENSOR_VARNAME = "rhd.irsensor";
032     private static final String ENCODER_LEFT_VARNAME = "rhd.enc1";
033     private static final String ENCODER_RIGHT_VARNAME = "rhd.encr";
034     private static final int LINESENSOR_ROOF = 128;
035     private static final int IRSENSOR_ROOF = 200;
036     private XMLClientHandler handler;
037     private LineGraph irGraph;
038     private LineGraph lineGraph;
039     private WheelControl leftWheel;
040     private WheelControl rightWheel;
041
042     /** Creates new form SMRPlugin */
043     public SMRPlugin() {
044         initComponents();
045         irGraph = new LineGraph();
046         lineGraph = new LineGraph();
047         jPanelIRGraph.add(irGraph, BorderLayout.CENTER);
048         jPanelLineGraph.add(lineGraph, BorderLayout.CENTER);
049         leftWheel = new WheelControl(this, WheelControl.Wheel.LEFT);
050         rightWheel = new WheelControl(this, WheelControl.Wheel.RIGHT);
051         jPanelLeftWheel.add(leftWheel, BorderLayout.CENTER);
052         jPanelRightWheel.add(rightWheel, BorderLayout.CENTER);
053     }
054
055     private void initComponents() {
056         // Removed auto-generated code
057
058         public String getPluginName() {
059             return "SMR";
060         }
061
062         public void setXMLClientHandler(XMLClientHandler handler) {
063             this.handler = handler;
064         }
065
066         public void doUpdate() {
067             //
068         }
069
070         public void startPlugin() {
071             handler.addPropertyChangeListener(VarDataPlugin.SUBSCRIBE_VARDATA, th
```

```

is);
213         //TODO: Push sensor data once each second.. Disabled for now
214         handler.sendCmd("push cmd=\"var rhd.linesensor\"");
215         handler.sendCmd("push cmd=\"var rhd.irsensor\"");
216         handler.sendCmd("push t=\"\"+ ENCODER_UPDATE_DELAY/1000 +\"\" cmd=\"var
rhd.encr\"");
217         handler.sendCmd("push t=\"\"+ ENCODER_UPDATE_DELAY/1000 +\"\" cmd=\"var
rhd.encl\"");
218     }
219
220     public void stopPlugin() {
221         leftWheel.stopUpdateThread();
222         rightWheel.stopUpdateThread();
223     }
224
225     public JPanel getJPanel() {
226         return this;
227     }
228
229     public void propertyChange(PropertyChangeEvent evt) {
230         if (evt.getPropertyName().equals(VarDataPlugin.SUBSCRIBE_VARDATA)) {
231             ModuleVariable modVar = (ModuleVariable) evt.getNewValue();
232             parseSensorVariables(modVar);
233             parseWheelEncoderVariables(modVar);
234             parseSpeedVariables(modVar);
235         }
236     }
237
238     public int[] getArrayFromSensorString(String value) {
239         String[] sValues = value.split(" ");
240         int[] data = new int[sValues.length - 1]; //-
1 since we don't include the first 1 in Line/IRSensor
241         for (int i = 1; i < sValues.length; i++) {
242             try {
243                 int number = Integer.parseInt(sValues[i]);
244                 data[i-1] = number;
245             } catch (NumberFormatException ex) {
246                 System.err.println("Failed to parse " + sValues[i]);
247             }
248         }
249         return data;
250     }
251     // Variables declaration - do not modify//GEN-BEGIN:variables
252     private javax.swing.JLabel jLabel1;
253     private javax.swing.JLabel jLabel2;
254     private javax.swing.JLabel jLabel3;
255     private javax.swing.JLabel jLabel4;
256     private javax.swing.JPanel jPanel1;
257     private javax.swing.JPanel jPanel2;
258     private javax.swing.JPanel jPanel3;
259     private javax.swing.JPanel jPanelIRGraph;
260     private javax.swing.JPanel jPanelLeftWheel;
261     private javax.swing.JPanel jPanelLineGraph;
262     private javax.swing.JPanel jPanelRightWheel;
263     // End of variables declaration//GEN-END:variables
264
265     public static void main(String[] args) {
266         JFrame f = new JFrame();
267         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
268         f.add(new SMRPlugin());
269         f.setSize(400, 400);
270         f.setLocation(200, 200);
271         f.setVisible(true);
272     }

```

```

273
274 private void parseSensorVariables(ModuleVariable modVar) {
275     if (modVar.getVarName().equals(LINESENSOR_VARNAME)) {
276         int[] data = getArrayFromSensorString(modVar.getValue());
277         lineGraph.setData(data, LINESENSOR_ROOF);
278     } else if (modVar.getVarName().equals(IRSENSOR_VARNAME)) {
279         int[] data = getArrayFromSensorString(modVar.getValue());
280         irGraph.setData(data, IRSENSOR_ROOF);
281     }
282 }
283
284 private void parseWheelEncoderVariables(ModuleVariable modVar) {
285     if (modVar.getVarName().equals(ENCODER_LEFT_VARNAME)) {
286         leftWheel.newEncoderValue(getValue(modVar));
287     } else if (modVar.getVarName().equals(ENCODER_RIGHT_VARNAME)) {
288         rightWheel.newEncoderValue(getValue(modVar));
289     }
290 }
291
292 private int getValue(ModuleVariable modVar) {
293     String[] valuesS = modVar.getValue().split(" ");
294     return Integer.parseInt(valuesS[1]);
295 }
296
297 public XMLClientHandler getXMLClientHandler() {
298     return handler;
299 }
300
301 private void parseSpeedVariables(ModuleVariable modVar) {
302     if (modVar.getVarName().equals("rhd.speedr")) {
303         int speed = getValue(modVar);
304         rightWheel.setSpeedLabel(speed);
305     } else if (modVar.getVarName().equals("rhd.speedl")) {
306         int speed = getValue(modVar);
307         leftWheel.setSpeedLabel(speed);
308     }
309 }
310 }

```


marg.gui.plugin.smr.LineGraph

```
005 package marg.gui.plugin.smr;
006
007 import java.awt.Color;
008 import java.awt.Font;
009 import java.awt.Graphics;
010 import java.awt.Graphics2D;
011 import java.awt.Rectangle;
012 import java.awt.RenderingHints;
013 import java.awt.font.FontRenderContext;
014 import javax.swing.JFrame;
015 import javax.swing.JPanel;
016 import javax.swing.SwingUtilities;
017
018 /**
019  *
020  * @author MARG
021  */
022 public class LineGraph extends JPanel {
023
024     private int[] data;
025     private int maxValue = 10;
026     private static final int PAD = 10;
027     private static final int LABEL_BAR_HEIGHT = 15;
028     private static final int SPACE_BETWEEN_BARS = 5;
029     private static final int AXIS_WIDTH = 2;
030     private Font font = new Font("Tahoma", Font.BOLD, 12);
031
032     public LineGraph() {
033         super();
034         this.setBackground(new Color(233, 233, 237));
035     }
036
037     public LineGraph(int[] data) {
038         super();
039         this.setBackground(new Color(233, 233, 237));
040         this.setData(data); //figures out maxValue itself
041     }
042
043     public LineGraph(int[] data, int maxValue) {
044         super();
045         this.setBackground(new Color(233, 233, 237));
046         this.setData(data, maxValue);
047     }
048
049     @Override
050     public void paintComponent(Graphics g) {
051         super.paintComponent(g);
052         paintBarGraph(g);
053     }
054
055     public void setData(int[] data, int maxValue) {
056         this.data = data;
057         this.maxValue = maxValue;
058         //System.out.printf("data = %s\n", java.util.Arrays.toString(data));
059         redrawLineGraph();
060     }
061
062     public void setMaxValue(int maxValue) {
063         this.maxValue = maxValue;
064     }
065
066     public void setData(int[] data) {
```

```

067         int highValue = 0;
068         for (int i : data) {
069             if (i > highValue) {
070                 highValue = i;
071             }
072         }
073         this.data = data;
074         this.maxValue = highValue + (int) (highValue * 0.2); //20% relative da
ta higher
075         //System.out.printf("data = %s\n", java.util.Arrays.toString(data));
076         redrawLineGraph();
077     }
078
079     protected void paintBarGraph(Graphics g) {
080         super.paintComponent(g);
081
082         Graphics2D g2 = (Graphics2D) g;
083         //Text Anti-aliasing: Disabled it for now as it didn't seem needed.
084         //g2.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING, RenderingH
ints.VALUE_TEXT_ANTIALIAS_ON);
085
086         int w = getWidth();
087         int h = getHeight();
088         g2.setPaint(new Color(50, 100, 133));
089         // Draw ordinate.
090         g2.fill(new Rectangle(PAD - AXIS_WIDTH, PAD, AXIS_WIDTH, h + 2 -
(PAD * 2) - LABEL_BAR_HEIGHT)); //+2 so that it crosses the abscissa
091         // Draw abscissa.
092         g2.fill(new Rectangle(PAD, h - PAD - LABEL_BAR_HEIGHT, w -
PAD * 2, AXIS_WIDTH));
093         if (data != null) {
094             double xInc = (double) (w - 2 * PAD) / data.length;
095             double scale = (double) (h - 2 * PAD) / maxValue;
096             // Draw data.
097             double x = PAD;
098             double y = h - PAD;
099             int barWidth = (int) xInc - SPACE_BETWEEN_BARS;
100             for (int i = 0; i < data.length; i++) {
101                 x = PAD + i * xInc;
102                 y = h - PAD - LABEL_BAR_HEIGHT - scale * data[i];
103                 g2.setPaint(new Color(180, 180, 233));
104                 double barHeightDouble = scale * data[i];
105                 int barHeight = (int) Math.ceil(barHeightDouble);
106                 g2.fill(new Rectangle((int) x + SPACE_BETWEEN_BARS, (int)
y, barWidth, barHeight));
107
108                 g2.setPaint(Color.BLACK);
109                 g2.setFont(font);
110                 FontRenderContext frc = g2.getFontRenderContext();
111                 float barLabelWidth = (float) font.getStringBounds(String.
valueOf(data[i]), frc).getWidth();
112                 float labelWidth = (float) font.getStringBounds(String.val
ueOf(i), frc).getWidth();
113                 //Draw Number
114                 g2.drawString(String.valueOf(data[i]), (int) x + SPACE_BET
WEEN_BARS + (barWidth / 2) - (barLabelWidth / 2), (int) y - 2);
115                 //Draw Label
116                 g2.drawString(String.valueOf(i+1), (int) x + SPACE_BETWEEN
_BARS + (barWidth / 2) - (labelWidth / 2), (int) h - LABEL_BAR_HEIGHT/2);
117             }
118         }
119     }
120
121     public static void main(String[] args) {

```

```

122         JFrame f = new JFrame();
123         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
124         int[] data = {169, 128, 90, 89, 2, 255};
125         int[] data2 = {1, 4, 8, 10, 2, 5};
126         LineGraph graph = new LineGraph(data, 200);
127         f.add(graph);
128         f.setSize(400, 400);
129         f.setLocation(200, 200);
130         f.setVisible(true);
131     }
132
133     private void redrawLineGraph() {
134         SwingUtilities.invokeLater(
135             new Runnable() {
136                 public void run() {
137                     LineGraph.this.repaint();
138                 }
139             });
140     }
141 }

```

marg.gui.plugin.smr.RobotWheel

```

05 package marg.gui.plugin.smr;
06
07 import java.awt.BorderLayout;
08 import java.awt.Graphics;
09 import java.awt.Graphics2D;
10 import java.awt.Image;
11 import java.awt.RenderingHints;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import javax.swing.JButton;
15 import javax.swing.JFrame;
16 import javax.swing.JPanel;
17 import marg.images.ResUtils;
18
19 /**
20  *
21  * @author MARG
22  */
23 public class RobotWheel extends JPanel {
24
25     private static final double WHEEL_SCALE = 0.5;
26     private Image wheelImg;
27     private double angle = 0;
28
29     public RobotWheel() {
30         super();
31         this.setOpaque(false);
32         wheelImg = ResUtils.getBufferedImage("SMRWheel.gif");
33     }
34
35     public void moveWheel(double degrees) {
36         double newAngle = angle + degrees;
37         if (newAngle > 360) {
38             angle = newAngle % 360;
39         } else if (newAngle < 0) {
40             angle = 360 + (newAngle % 360);
41         } else {
42             angle = newAngle;
43         }
44         //System.out.println("Angle is " + angle);

```

```

45         this.repaint();
46     }
47
48     @Override
49     protected void paintComponent(Graphics g) {
50         super.paintComponent(g);
51         Graphics2D g2 = (Graphics2D) g;
52         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
53         int centerX = this.getWidth() / 2;
54         int centerY = this.getHeight() / 2;
55         int x = centerX - (wheelImg.getWidth(null) / 4);
56         int y = centerY - (wheelImg.getHeight(null) / 4);
57
58         int width = (int) (wheelImg.getWidth(null) * WHEEL_SCALE);
59         int height = (int) (wheelImg.getHeight(null) * WHEEL_SCALE);
60
61         int wheelX = x + (wheelImg.getWidth(null) / 4);
62         int wheelY = y + (wheelImg.getHeight(null) / 4);
63         //System.out.println("Wheel angle is " + angle);
64
65         //g2.drawString("X", wheelX, wheelY); //Draws X on center
66         g2.rotate(angle * Math.PI / 180.0, wheelX, wheelY);
67         g2.drawImage(wheelImg, x, y, width, height, null);
68         //g2.drawImage(wheelImg, null, null);
69     }
70
71     public static void main(String[] args) {
72         JFrame f = new JFrame();
73         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
74         final RobotWheel robWheel = new RobotWheel();
75         f.setLayout(new BorderLayout());
76         f.add(robWheel, BorderLayout.CENTER);
77         f.setSize(400, 400);
78         f.setLocation(200, 200);
79         f.setVisible(true);
80         JButton test = new JButton("Test Forward");
81         test.setSize(100, 20);
82         f.add(test, BorderLayout.SOUTH);
83         test.addActionListener(new ActionListener() {
84             public void actionPerformed(ActionEvent e) {
85                 robWheel.moveWheel(-10);
86                 robWheel.repaint();
87             }
88         });
89     }
90 }

```

marg.gui.plugin.smr.WheelControl

```
011 package marg.gui.plugin.smr;
012
013 import java.awt.BorderLayout;
014 import java.util.logging.Level;
015 import java.util.logging.Logger;
016 import javax.swing.JFrame;
017 import javax.swing.JSlider;
018 import javax.swing.event.ChangeEvent;
019 import javax.swing.event.ChangeListener;
020 import marg.gui.plugin.SMRPlugin;
021
022 /**
023  *
024  * @author MARG
025  */
026 public class WheelControl extends javax.swing.JPanel implements ChangeListener
, Runnable {
027
028     public enum Wheel {
029         RIGHT,
030         LEFT
031     };
032
033     private static final double ENCODER_PER_ANGLE = 2000 / 360.0;
034     private SMRPlugin parent;
035     private RobotWheel robWheel;
036     private Wheel wheel;
037     private boolean threadRunning = false;
038     private int lastEncoderValue = Integer.MAX_VALUE;
039     private int lastSentSpeed = Integer.MAX_VALUE;
040     private int encoderSpeed = 0;
041
042     /** Creates new form WheelControl */
043     public WheelControl(SMRPlugin parent, Wheel wheel) {
044         initComponents();
045         this.wheel = wheel;
046         this.parent = parent;
047         startUpdateThread();
048         robWheel = new RobotWheel();
049         jPanelWheel.add(robWheel, BorderLayout.CENTER);
050         jSlider1.addChangeListener(this);
051     }
052
053     /** This method is called from within the constructor to
054      * initialize the form.
055      * WARNING: Do NOT modify this code. The content of this method is
056      * always regenerated by the Form Editor.
057      */
058     @SuppressWarnings("unchecked")
059     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN: initComponents
060     private void initComponents() {
114     } // Removed auto-generated code
121
122     public static void main(String[] args) {
123         JFrame f = new JFrame();
124         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
125         WheelControl wheelControl = new WheelControl(null, Wheel.LEFT);
126         f.add(wheelControl);
127         f.setSize(200, 250);
128         f.setLocation(200, 200);
129         f.setVisible(true);
```

```

130         wheelControl.newEncoderValue(0);
131         wheelControl.newEncoderValue(-2000);
132     }
133
134     private void startUpdateThread() {
135         Thread t = new Thread(this);
136         threadRunning = true;
137         t.start();
138     }
139
140     public void newEncoderValue(int newValue) {
141         if (lastEncoderValue == Integer.MAX_VALUE) {
142             lastEncoderValue = newValue;
143         } else {
144             //TODO: Encoder Overflow, enc tallet skifter 7FFF + 1 = -
145             7FFE (når den kører 1 tick fremad og får stod på 7FFF)
146             int encDiff = newValue - lastEncoderValue;
147             //System.out.println("Encoder Diff: " + encDiff);
148             encoderSpeed = encDiff;
149             lastEncoderValue = newValue;
150         }
151         jLabelEncoder.setText("Encr: " + newValue);
152     }
153
154     public void run() {
155         while (threadRunning) {
156             double changeAngle = encoderSpeed / ENCODER_PER_ANGLE;
157             //System.out.println("Change Angle: " + changeAngle);
158             if (changeAngle < 0 || changeAngle > 0) {
159                 robWheel.moveWheel(changeAngle / 20);
160             }
161             try {
162                 int delay = SMRPlugin.ENCODER_UPDATE_DELAY / 20;
163                 Thread.sleep(delay);
164             } catch (InterruptedException ex) {
165                 Logger.getLogger(WheelControl.class.getName()).log(Level.SEVERE,
166                 null, ex);
167             }
168         }
169     }
170
171     public void setSpeedLabel(int speed) {
172         jLabelSpeed.setText("Speed: " + speed + " cm/sec");
173     }
174
175     public void stateChanged(ChangeEvent e) {
176         JSlider source = (JSlider) e.getSource();
177         int sliderValue = source.getValue();
178         if (!source.getValueIsAdjusting() && sliderValue != lastSentSpeed) {
179             if (parent != null) {
180                 String speedVarName = "rhd.speedr";
181                 if (wheel == Wheel.LEFT)
182                     speedVarName = "rhd.speedl";
183                 parent.getXMLClientHandler().sendCmd("var " + speedVarName + "[1
184                 ]=\""+ sliderValue + "\"");
185                 lastSentSpeed = sliderValue;
186             }
187         }
188     }
189
190     public void stopUpdateThread() {
191         threadRunning = false;
192     }
193 }

```

marg.handlers.XMLClientHandler

```
06 package marg.handlers;
07
08 import java.beans.PropertyChangeListener;
09 import marg.model.XMLClient;
10 import marg.model.plugin.XMLParsePlugin;
11 import marg.model.XMLParser;
12
13 /**
14  *
15  * @author MARG
16  */
17 public class XMLClientHandler {
18
19     private static final String SETVALUE_CMD = "var %s=\"%s\"";
20     private XMLClient client;
21
22     public XMLClientHandler(XMLClient client) {
23         this.client = client;
24     }
25
26     public boolean connect(String host, int port) {
27         if (!client.isConnected())
28             return client.connect(host, port);
29         else
30             return false;
31     }
32
33     public void disconnect() {
34         client.disconnect();
35     }
36
37     public void addParsePlugin(XMLParsePlugin plugin) {
38         client.getXMLParser().addParsePlugin(plugin);
39     }
40
41     public void addPropertiesChangeListener(PropertyChangeListener listener, String... propertyNames) {
42         for (String propName : propertyNames) {
43             client.getXMLParser().addPropertyChangeListener(propName, listener);
44         }
45     }
46
47     public void addPropertyChangeListener(String propertyName, PropertyChangeListener listener) {
48         client.getXMLParser().addPropertyChangeListener(propertyName, listener);
49     }
50
51     public void removePropertyChangeListener(String propertyName, PropertyChangeListener listener) {
52         client.getXMLParser().removePropertyChangeListener(propertyName, listener);
53     }
54
55     public void sendCmd(String cmd) {
56         if (client.isConnected()) {
57             client.sendCmd(cmd);
58         } else {
59             System.err.println("Not Connected; Tried sending cmd: " + cmd);
60         }
61     }
62 }
```

```

62
63     public void setVariable(String varName, String newValue) {
64         String setValCmd = String.format(SETVALUE_CMD, varName, newValue);
65         client.sendCmd(setValCmd);
66         client.sendCmd("var allcopy");
67     }
68
69     public boolean isConnected() {
70         return client.isConnected();
71     }
72
73     public XMLParser getXMLParser() {
74         return client.getXMLParser();
75     }
76
77 }

```

marg.images.ResUtils

```

05 package marg.images;
06
07 import java.awt.image.BufferedImage;
08 import java.net.URL;
09 import java.util.logging.Logger;
10 import javax.imageio.ImageIO;
11 import javax.swing.ImageIcon;
12
13 /**
14  *
15  * @author MARG
16  */
17 public class ResUtils {
18
19     private static final String PATH = "/" + (ResUtils.class.getPackage().getNa
me().replace(".", "/")) + "/";
20
21     public static URL getURL(String name) {
22         return ResUtils.class.getResource(PATH + name);
23     }
24
25     public static ImageIcon getImageIcon(String name) {
26         return new ImageIcon(getBufferedImage(name));
27     }
28
29     /**
30      * Finds, reads and returns an image that can be accessed by class code in
a way that is independent of the location of the code.
31      * The name of a resource is a '/'-
separated path name that identifies the resource. Example: "marg/images/Wheel.gif"
32      * @param name The / separated name of the ressource to locate the image at
33      * @return the BufferedImage constructed from the given ressource. Returns
a 20x20 black placeholder if no ressource was found.
34      */
35     public static BufferedImage getBufferedImage(String name) {
36         try {
37             URL url = getURL(name);
38             marg.util.Log.GlobalLogger.info("Loaded image from "+ url);
39             return ImageIO.read(url);
40         } catch (Exception e) {
41             e.printStackTrace();
42             return new BufferedImage(20, 20, BufferedImage.TYPE_INT_RGB);
43         }
44     }

```



```

45
46     public static void main(String[] args) {
47         ResUtils.getImageIcon("GreenButton20px.gif");
48     }
49 }

```

marg.model.AbstractModuleClient

```

01 package marg.model;
02
03 import java.io.IOException;
04 import java.io.PrintWriter;
05 import java.net.ConnectException;
06 import java.net.InetSocketAddress;
07 import java.net.Socket;
08 import java.net.UnknownHostException;
09 import java.util.logging.Level;
10 import java.util.logging.Logger;
11
12 public abstract class AbstractModuleClient implements ModuleClient, Runnable {
13
14     private static final int SOCKET_CONNECT_TIMEOUT = 1500;
15     Socket socket;
16     volatile boolean threadRunning = false;
17     volatile boolean connected = false;
18     PrintWriter out;
19
20     public void disconnect() {
21         try {
22             connected = false;
23             threadRunning = false;
24             closeInputStream();
25             if (socket != null) {
26                 socket.close();
27             }
28         } catch (IOException ex) {
29             Logger.getLogger(AbstractModuleClient.class.getName()).log(Level.SEVERE, null, ex);
30         }
31     }
32
33     public boolean connect(String host, int port) {
34         try {
35             if (port < 1024 || port > 65535)
36                 throw new IllegalArgumentException();
37             socket = new Socket();
38             socket.bind(null);
39             socket.connect(new InetSocketAddress(host, port), SOCKET_CONNECT_TIMEOUT);
40             out = new PrintWriter(socket.getOutputStream());
41             openInputStream();
42             connected = true;
43             if (!threadRunning) {
44                 startUpdateThread();
45             }
46             return true;
47         } catch (ConnectException ex) {
48             marg.util.Log.GlobalLogger.warning("Failed to connect to: " + host + ":" + port);
49         } catch (UnknownHostException ex) {
50             marg.util.Log.GlobalLogger.warning("Could not resolve given host: " + ex.getMessage());
51         } catch (IOException ex) {

```

```

52         Logger.getLogger(AbstractModuleClient.class.getName()).log(Level.SEVERE, null, ex);
53     }
54     return false;
55 }
56
57 public void sendCmd(String cmd) {
58     if (out != null) {
59         out.println(cmd);
60     }
61 }
62
63 private void startUpdateThread() {
64     threadRunning = true;
65     Thread t = new Thread(this);
66     t.start();
67 }
68
69 public boolean isConnected() {
70     return connected;
71 }
72
73 public abstract void run();
74
75 abstract void openInputStream();
76 abstract void closeInputStream();
77 }

```

marg.model.ModuleClient

```

01 package marg.model;
02
03
04 public interface ModuleClient {
05
06     public void disconnect();
07
08     public boolean connect(String host, int port);
09
10     public void sendCmd(String cmd);
11 }

```

marg.model.ModuleVariable

```
06 package marg.model;
07
08 /**
09  *
10  * @author MARG
11  */
12 public class ModuleVariable {
13
14     private String varName;
15     private String varType;
16     private String value;
17
18     public ModuleVariable(String varName, String varType, String value) {
19         this.varName = varName;
20         this.varType = varType;
21         this.value = value;
22     }
23
24     public String getValue() {
25         return value;
26     }
27
28     public String getVarName() {
29         return varName;
30     }
31
32     public String getShortVarName() {
33         int offset = varName.lastIndexOf('.');
34         return varName.substring(offset+1);
35     }
36
37     public String getVarType() {
38         return varType;
39     }
40
41     public void setValue(String value) {
42         this.value = value;
43     }
44
45     public String toString() {
46         return "(" + varType + ") " + getShortVarName() + " = " + value;
47     }
48
49     @Override
50     public boolean equals(Object obj) {
51         if (obj == null) {
52             return false;
53         }
54         if (getClass() != obj.getClass()) {
55             return false;
56         }
57         final ModuleVariable other = (ModuleVariable) obj;
58         if ((this.varName == null) ? (other.varName != null) : !this.varName.equals(other.varName)) {
59             return false;
60         }
61         if ((this.varType == null) ? (other.varType != null) : !this.varType.equals(other.varType)) {
62             return false;
63         }
64         if (this.value != other.value && (this.value == null || !this.value.equals(other.value))) {
```

```

65         return false;
66     }
67     return true;
68 }
69
70 }

```

marg.model.MonitoredVariable

```

06 package marg.model;
07
08 import java.util.IllegalFormatException;
09
10 /**
11  *
12  * @author MARG
13  */
14 public class MonitoredVariable {
15
16     private ModuleVariable moduleVar;
17     private String presentation;
18
19     public MonitoredVariable(ModuleVariable var, String presentation) {
20         this.moduleVar = var;
21         this.presentation = presentation;
22     }
23
24     public String getPresentation() {
25         return presentation;
26     }
27
28     public ModuleVariable getModuleVar() {
29         return moduleVar;
30     }
31
32     public void setPresentation(String presentation) {
33         this.presentation = presentation;
34     }
35
36     @Override
37     public String toString() {
38         String presentedModVar = "!! Formatting Failed !!";
39         try {
40             presentedModVar = String.format(presentation, moduleVar.getValue())
41         ;
42         } catch (IllegalFormatException e) {
43             System.out.println("Formatting of MonitoredVariable failed");
44         }
45         return presentedModVar;
46     }
47 }

```

marg.model.StatusVariable

```
06 package marg.model;
07
08 import marg.util.Log;
09
10 /**
11  *
12  * @author MARG
13  */
14 public class StatusVariable {
15
16     private ModuleVariable var;
17     private ValueLimit limit;
18
19     public StatusVariable(ModuleVariable var) {
20         this.var = var;
21     }
22
23     public StatusVariable(ModuleVariable var, ValueLimit limit) {
24         this.var = var;
25         this.limit = limit;
26     }
27
28     public void setLimit(ValueLimit limit) {
29         this.limit = limit;
30     }
31
32     public ModuleVariable getModuleVar() {
33         return var;
34     }
35
36     public boolean isStatusCorrect() {
37         if (limit != null && var.getVarType().equals("d")) { //d represented as
Double
38             try {
39                 Double value = Double.parseDouble(var.getValue());
40                 return limit.checkCorrect(value);
41             } catch (NumberFormatException ex) {
42                 Log.GlobalLogger.info("Could not parse variable with type (d)\n
"+ ex);
43                 return false;
44             }
45         } else
46             return true;
47     }
48
49     public ValueLimit getLimit() {
50         return limit;
51     }
52
53     @Override
54     public String toString() {
55         String out = "";
56         if (isStatusCorrect()) {
57             out = var.getVarName() + " (" + var.getVarType() + ")" + " = " + var.get
Value();
58         } else {
59             out = var.getVarName() + " (" + var.getVarType() + ")" + " = " + var.get
Value() + " (Outside Value Limit)";
60         }
61         return out;
62     }
63 }
```

marg.model.ValueLimit

```
05 package marg.model;
06
07 /**
08  *
09  * @author MARG
10  */
11 public class ValueLimit {
12
13     public final static int LESS_THAN_EQUALS = 1;
14     public final static int LESS_THAN = 2;
15     private int minimumComparison;
16     private int maximumComparison;
17     private double maxValue;
18     private double minValue;
19
20     public ValueLimit(double minValue, int minComparison, double maxValue, int
maxComparison) {
21         this.maxValue = maxValue;
22         this.minValue = minValue;
23         minimumComparison = minComparison;
24         maximumComparison = maxComparison;
25     }
26
27     public boolean checkCorrect(double value) {
28         boolean minCorrect = false;
29         boolean maxCorrect = false;
30         if (minimumComparison == LESS_THAN_EQUALS) {
31             minCorrect = minValue <= value;
32         } else {
33             minCorrect = minValue < value;
34         }
35         if (maximumComparison == LESS_THAN_EQUALS) {
36             maxCorrect = maxValue >= value;
37         } else {
38             maxCorrect = maxValue > value;
39         }
40         boolean isCorrect = (minCorrect && maxCorrect);
41         return isCorrect;
42     }
43
44     public double getMaxValue() {
45         return maxValue;
46     }
47
48     public double getMinValue() {
49         return minValue;
50     }
51
52     public int getMaximumComparison() {
53         return maximumComparison;
54     }
55
56     public int getMinimumComparison() {
57         return minimumComparison;
58     }
59
60     public String toString() {
61         return "ValueLimit (min "+ minValue +", max "+ maxValue +", MinLTE "+ (
```

```

minimumComparison == LESS_THAN_EQUALS) + ", MaxLTE " + (maximumComparison == LESS_THAN_EQUALS) + " ";
62     }
63 }

```

marg.model.XMLClient

```

01 package marg.model;
02
03 import java.io.IOException;
04 import java.util.logging.Level;
05 import java.util.logging.Logger;
06 import marg.util.Log;
07 import org.xml.sax.InputSource;
08 import org.xml.sax.SAXException;
09 import org.xml.sax.XMLReader;
10 import org.xml.sax.helpers.XMLReaderFactory;
11
12 public class XMLClient extends AbstractModuleClient {
13
14     private XMLReader xr;
15     private XMLParser xp;
16     private InputSource in;
17     private static final String NAMESPACE = "marg";
18
19     public XMLClient() {
20         try {
21             xr = XMLReaderFactory.createXMLReader();
22             xp = new XMLParser();
23             xr.setContentHandler(xp);
24         } catch (SAXException ex) {
25             Logger.getLogger(XMLClient.class.getName()).log(Level.SEVERE, null,
26 ex);
27         }
28
29         @Override
30         public void run() {
31             Log.GlobalLogger.fine("Run Thread Started");
32             while (threadRunning) {
33                 try {
34                     if (connected) {
35                         Log.GlobalLogger.info("Socket is connected, parsing XML");
36                         xr.parse(in);
37                     } else {
38                         Thread.sleep(200);
39                     }
40                 } catch (IOException ex) {
41                     //Logger.getLogger(XMLClient.class.getName()).log(Level.SEVERE,
42 null, ex);
43                     System.err.println("IOException, Socket closed while parsing");
44                     connected = false;
45                 } catch (SAXException ex) {
46                     Logger.getLogger(XMLClient.class.getName()).log(Level.SEVERE, n
47 ull, ex);
48                     System.out.println("SAX xr.parse exit");
49                     connected = false;
50                 } catch (InterruptedException ex) {
51                     Logger.getLogger(XMLClient.class.getName()).log(Level.SEVERE, n
52 ull, ex);
53                     System.out.println("Interrupted while sleeping");
54                 }
55             }
56         }
57     }
58 }

```

```

55     @Override
56     public void sendCmd(String cmd) {
57         if (out != null) {
58             String xmlCmd = xmlFormatCmd(cmd);
59             Log.GlobalLogger.info("Sending cmd: " + xmlCmd);
60             out.println(xmlCmd);
61             out.flush();
62         }
63     }
64
65     public String xmlFormatCmd(String cmd) {
66         return "<" + cmd + ">";
67     }
68
69     @Override
70     void openInputStream() {
71         try {
72             in = new InputSource(socket.getInputStream());
73             //Declare XML Language and namespace
74             out.println("<?xml version=\"1.0\" encoding=\"UTF-8\" ?>");
75             out.println("<" + NAMESPACE + " name=\"XMLClient\" version=\"1.0\">");
76         } catch (IOException ex) {
77             Logger.getLogger(XMLClient.class.getName()).log(Level.SEVERE, null,
78                 ex);
79         }
80
81         @Override
82         void closeInputStream() {
83             in = null;
84             //Close namespace
85             if (out != null) {
86                 out.println("</" + NAMESPACE + ">");
87             }
88         }
89
90         public XMLParser getXMLParser() {
91             return xp;
92         }
93     }

```

marg.model.XMLClientGUI

```

012 package marg.model;
013
014 import java.beans.PropertyChangeEvent;
015 import java.beans.PropertyChangeListener;
016
017 /**
018  *
019  * @author MARG
020  */
021 public class XMLClientGUI extends javax.swing.JApplet implements PropertyChangeListener {
022
023     XMLClient client;
024
025     /** Initializes the applet XMLClientGUI */
026     public void init() {
027         try {
028             java.awt.EventQueue.invokeAndWait(new Runnable() {
029                 public void run() {

```



```

030         initComponents();
031     }
032     });
033     } catch (Exception ex) {
034         ex.printStackTrace();
035     }
036 }
037
045 private void initComponents() {
154     // Removed auto-generated code
155
156     private void hostTFActionPerformed(java.awt.event.ActionEvent evt)
157         // TODO add your handling code here:
158     }
159
160     private void connectBTNActionPerformed(java.awt.event.ActionEvent evt) {
161         client = new XMLClient();
162         client.connect(hostTF.getText(), Integer.parseInt(portTF.getText()));
163         client.getXMLParser().addPropertyChangeListener("rawData", this);
164     }
165
166     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
167         client.sendCmd(cmdTF.getText());
168     }
185     public void propertyChange(PropertyChangeEvent evt) {
186         String newRawData = (String)evt.getNewValue();
187         System.out.println(newRawData);
188         jTextArea1.append(newRawData);
189     }
190
191 }

```

marg.model.XMLParser

```

05 package marg.model;
06
07 import marg.model.plugin.XMLParsePlugin;
08 import java.beans.PropertyChangeListener;
09 import java.beans.PropertyChangeSupport;
10 import java.util.ArrayList;
11 import org.xml.sax.Attributes;
12 import org.xml.sax.SAXException;
13 import org.xml.sax.helpers.DefaultHandler;
14
15 /**
16  *
17  * @author MARG
18  */
19 public class XMLParser extends DefaultHandler {
20
21     private String currentTag;
22     XMLParsedData data;
23
24     public XMLParser() {
25         this.data = new XMLParsedData();
26     }
27
28     private ArrayList<XMLParsePlugin> parsePlugins = new ArrayList<XMLParsePlug
in>();
29     PropertyChangeSupport prop = new PropertyChangeSupport(this);
30
31     public void addParsePlugin(XMLParsePlugin plugin) {
32         plugin.setPropertyChangeSupport(prop);
33         parsePlugins.add(plugin);
34     }

```

```

35
36     public void removeParsePlugin(XMLParsePlugin plugin) {
37         parsePlugins.remove(plugin);
38     }
39
40     public void addPropertyChangeListener(String propertyName, PropertyChangeLi
stener listener) {
41         prop.addPropertyChangeListener(propertyName, listener);
42     }
43
44     public void removePropertyChangeListener(String propertyName, PropertyChang
eListener listener) {
45         prop.removePropertyChangeListener(propertyName, listener);
46     }
47
48     public void startDocument() throws SAXException {
49         //System.out.println("Started Document");
50     }
51
52     public void endDocument() throws SAXException {
53         //System.out.println("Ended Document");
54     }
55
56     public void startElement(String uri, String localName,
57         String qName, Attributes attributes)
58         throws SAXException {
59         currentTag = qName;
60         for (XMLParsePlugin plugin : parsePlugins) {
61             plugin.startElement(qName, attributes);
62         }
63     }
64
65     public void endElement(String uri, String localName, String qName)
66         throws SAXException {
67         for (XMLParsePlugin plugin : parsePlugins) {
68             plugin.endElement(qName);
69         }
70     }
71
72     public void characters(char ch[], int start, int length)
73         throws SAXException {
74         for (XMLParsePlugin plugin : parsePlugins) {
75             plugin.tagContents(currentTag, ch, start, length);
76         }
77     }
78 }

```

marg.model.plugin.XMLParsePlugin

```
06 package marg.model.plugin;
07
08 import java.beans.PropertyChangeSupport;
09 import org.xml.sax.Attributes;
10
11
12 /**
13  *
14  * @author MARG
15  */
16 public interface XMLParsePlugin {
17
18     /**
19      * Called when the starttag of an element is received
20      * @param tagName name of the element/tag
21      * @param atts attributes of the element
22      */
23     public void startElement(String tagName, Attributes atts);
24
25     /**
26      * Called when the endtag of an element is received
27      * @param tagName name of the element/tag
28      */
29     public void endElement(String tagName);
30
31     /**
32      * Called when contents of an element is received. Data can be of any kind
33      * @param parentTag name of the first parent tag this content was wrapped i
34      n
35      * @param ch character buffer
36      * @param start start offset for the content in character buffer
37      * @param length length of content
38      */
39     public void tagContents(String parentTag, char ch[], int start, int length)
40     ;
41
42     /**
43      * Sets the PropertyChangeSupport object of the parent XMLParser.
44      * By contract this method will be called as the first thing when an XMLPar
45      sePlugin is added.
46      * This is a central callback hub that should be used to fire PropertyChang
47      eEvents.
48      * This is the standard way to make callbacks when new data is received.
49      * Data can be sent as the new value when you fire a callback,
50      * but feel free to make your own data access methods as well
51      * @param prop the PropertyChangeSupport object sent from parent XMLParser
52      */
53     public void setPropertyChangeSupport(PropertyChangeSupport prop);
54 }
```

marg.model.plugin.RawDataPlugin

```
05 package marg.model.plugin;
06
07 import java.beans.PropertyChangeSupport;
08 import org.xml.sax.Attributes;
09
10 /**
11  *
12  * @author MARG
13  */
14 public class RawDataPlugin implements XMLParsePlugin {
15
16     public final static String SUBSCRIBE_RAWDATA = "rawData";
17     private PropertyChangeSupport prop;
18     private String rawData;
19
20     public RawDataPlugin() {
21     }
22
23     public void setPropertyChangeSupport(PropertyChangeSupport prop) {
24         this.prop = prop;
25     }
26
27     public void startElement(String tagName, Attributes atts) {
28         String oldRaw = rawData;
29         StringBuilder sb = new StringBuilder();
30         sb.append("<");
31         sb.append(tagName);
32         //System.out.println("Start Element: " + tagName);
33         //System.out.println("Attributes:");
34         for (int i = 0; i < atts.getLength(); i++) {
35             //System.out.println("\t" + atts.getQName(i) + " = " + atts.getValue
e(i));
36             sb.append(" ");
37             sb.append(atts.getQName(i) + "=\"" + atts.getValue(i) + "\"");
38         }
39         sb.append(">");
40         rawData = sb.toString();
41         prop.firePropertyChange(SUBSCRIBE_RAWDATA, oldRaw, rawData);
42     }
43
44     public void endElement(String tagName) {
45         String oldRaw = rawData;
46         rawData = "</" + tagName + ">";
47         prop.firePropertyChange(SUBSCRIBE_RAWDATA, oldRaw, rawData);
48     }
49
50     public void tagContents(String parentTag, char ch[], int start, int length)
51     {
52         String contents = new String(ch, start, length);
53         prop.firePropertyChange(SUBSCRIBE_RAWDATA, "", contents);
54     }
```

marg.model.plugin.VarDataPlugin

```
06 package marg.model.plugin;
07
08 import java.beans.PropertyChangeSupport;
09 import marg.model.ModuleVariable;
10 import org.xml.sax.Attributes;
11
12 /**
13  *
14  * @author MARG
15  */
16 public class VarDataPlugin implements XMLParsePlugin {
17
18     public final static String SUBSCRIBE_VARDATA = "varData";
19     private PropertyChangeSupport prop;
20     private ModuleVariable lastVariable;
21
22     public VarDataPlugin() {
23     }
24
25     public void setPropertyChangeSupport(PropertyChangeSupport prop) {
26         this.prop = prop;
27     }
28
29     public void startElement(String tagName, Attributes atts) {
30         if (tagName.equals("var") && atts.getLength() > 2) {
31             ModuleVariable oldVar = lastVariable;
32             String varName = atts.getValue("name");
33             String varType = atts.getValue("typ");
34             String varValue = atts.getValue("value");
35             ModuleVariable newVar = new ModuleVariable(varName, varType, varValue);
36             lastVariable = newVar;
37             prop.firePropertyChange(SUBSCRIBE_VARDATA, oldVar, newVar);
38         }
39     }
40
41     public void endElement(String tagName) {
42     }
43
44     public void tagContents(String parentTag, char ch[], int start, int length)
45     {
46     }
47 }
```

marg.test.mockserver.JUnitMockServer

```
06 package marg.test.mockserver;
07
08 import java.io.BufferedReader;
09 import java.io.IOException;
10 import java.io.InputStreamReader;
11 import java.io.PrintWriter;
12 import java.net.ServerSocket;
13 import java.net.Socket;
14 import java.util.Date;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17
18 /**
19  *
20  * @author MARG
21  */
22 public class JUnitMockServer implements Runnable {
23
24     private String lastReceivedCmd = "";
25     private String rootElement = "<mockServer>";
26     private String dummyAnswer = "<var name=\"core.version\" typ=\"d\" value=\"
2.04\"/>";
27     private int port;
28
29     public JUnitMockServer(int port) {
30         this.port = port;
31         Thread t = new Thread(this);
32         t.start();
33     }
34
35     public void run() {
36         try {
37             String sentence;
38             ServerSocket serverSocket = new ServerSocket(port);
39             while (true) {
40                 System.out.println("Server Started: Listening on port " + port)
;
41                 Socket socket = serverSocket.accept();
42                 PrintWriter outToClient = new PrintWriter(socket.getOutputStream(), true);
43                 BufferedReader inFromClient = new BufferedReader(new InputStreamReader(socket.getInputStream()));
44                 System.out.println("Connection OPENED. " + new Date());
45                 System.out.println("Client IP: " + socket.getInetAddress());
46                 outToClient.println(rootElement);
47                 while ((sentence = inFromClient.readLine()) != null) {
48                     System.out.println("Received cmd: " + sentence);
49                     lastReceivedCmd = sentence;
50                     outToClient.println(dummyAnswer);
51                     outToClient.flush();
52                 }
53                 System.out.println("Connection CLOSED.");
54                 socket.close();
55             }
56         } catch (IOException ex) {
57             Logger.getLogger(MockServer.class.getName()).log(Level.SEVERE, null
, ex);
58         }
59     }
60
61     public void setDummyAnswer(String dummyAnswer) {
62         this.dummyAnswer = dummyAnswer;
63     }
64 }
```

```

63     }
64
65     public String getLastReceivedCmd() {
66         return lastReceivedCmd;
67     }
68
69     public static void main(String[] args) {
70         MockServer mock = new MockServer(24928);
71     }
72 }

```

marg.test.mockserver.MockAllCopyServer

```

005 package marg.test.mockserver;
006
007 import java.io.BufferedReader;
008 import java.io.File;
009 import java.io.FileInputStream;
010 import java.io.FileNotFoundException;
011 import java.io.IOException;
012 import java.io.InputStream;
013 import java.io.InputStreamReader;
014 import java.io.OutputStream;
015 import java.io.PrintWriter;
016 import java.net.ServerSocket;
017 import java.net.Socket;
018 import java.net.SocketException;
019 import java.util.Date;
020 import java.util.logging.Level;
021 import java.util.logging.Logger;
022
023 /**
024  *
025  * @author MARG
026  */
027 public class MockAllCopyServer implements Runnable {
028
029     private String lastReceivedCmd = "";
030     private String rootElement = "mockServer";
031     private String dummyAnswer = "<var name=\"core.test.deep.struct.version\"
typ=\"d\" value=\"2.04\"/>";
032     private int port;
033
034     public MockAllCopyServer(int port) {
035         this.port = port;
036         Thread t = new Thread(this);
037         t.start();
038     }
039
040     public void run() {
041         try {
042             String sentence;
043             ServerSocket serverSocket = new ServerSocket(port);
044             while (true) {
045                 System.out.println("Server Started: Listening on port " + port
);
046                 Socket socket = serverSocket.accept();
047                 PrintWriter outToClient = new PrintWriter(socket.getOutputStream(), true);
048                 BufferedReader inFromClient = new BufferedReader(new InputStreamReader(socket.getInputStream()));
049                 System.err.println("Connection OPENED. " + new Date());
050                 System.out.println("Client IP: " + socket.getInetAddress());

```

```

051         outToClient.println("<" + rootElement + ">");
052
053         try {
054             File allCopyFile1 = new File("var_allcopyd1.xml"); //relative path
055             File allCopyFile2 = new File("var_allcopyd2.xml"); //relative path
056             System.out.println(allCopyFile1.getAbsolutePath());
057             while ((sentence = inFromClient.readLine()) != null && !sentence.equals("<quit/>")) {
058                 System.out.println("Received cmd: " + sentence);
059                 lastReceivedCmd = sentence;
060                 sendFileContents(allCopyFile1, socket.getOutputStream());
061                 try {
062                     Thread.sleep(1000);
063                 } catch (InterruptedException ex) {
064                     Logger.getLogger(MockAllCopyServer.class.getName()).log(Level.SEVERE, null, ex);
065                 }
066                 sendFileContents(allCopyFile2, socket.getOutputStream());
067             }
068             outToClient.println("</" + rootElement + ">");
069             System.err.println("Connection CLOSED.");
070             socket.close();
071         } catch (SocketException ex) {
072         } finally {
073             if (socket != null) {
074                 socket.close();
075             }
076         }
077     }
078 } catch (IOException ex) {
079     Logger.getLogger(MockAllCopyServer.class.getName()).log(Level.SEVERE, null, ex);
080 }
081 }
082
083 public void setDummyAnswer(String dummyAnswer) {
084     this.dummyAnswer = dummyAnswer;
085 }
086
087 public String getLastReceivedCmd() {
088     return lastReceivedCmd;
089 }
090
091 public static void main(String[] args) {
092     MockAllCopyServer mock = new MockAllCopyServer(24930);
093 }
094
095 public void sendFileContents(File file, OutputStream out) {
096     try {
097         FileInputStream inFile = new FileInputStream(file);
098         copyContents(inFile, out);
099     } catch (FileNotFoundException ex) {
100         Logger.getLogger(MockAllCopyServer.class.getName()).log(Level.SEVERE, null, ex);
101     }
102 }
103
104 public void copyContents(InputStream from, OutputStream to) {
105     try {
106         byte[] buffer = new byte[512];

```



```

107         int bytesRead = 0;
108         while ((bytesRead = from.read(buffer)) != -1) {
109             to.write(buffer, 0, bytesRead);
110             buffer = new byte[512];
111         }
112         to.flush();
113     } catch (IOException ex) {
114         Logger.getLogger(MockAllCopyServer.class.getName()).log(Level.SEVERE, null, ex);
115     }
116 }
117 }

```

marg.test.mockserver.MockServer

```

005 package marg.test.mockserver;
006
007 import java.io.BufferedReader;
008 import java.io.File;
009 import java.io.FileInputStream;
010 import java.io.FileNotFoundException;
011 import java.io.IOException;
012 import java.io.InputStream;
013 import java.io.InputStreamReader;
014 import java.io.OutputStream;
015 import java.io.PrintWriter;
016 import java.net.ServerSocket;
017 import java.net.Socket;
018 import java.net.SocketException;
019 import java.util.Date;
020 import java.util.logging.Level;
021 import java.util.logging.Logger;
022
023 /**
024  *
025  * @author MARG
026  */
027 public class MockServer implements Runnable {
028
029     private String lastReceivedCmd = "";
030     private String rootElement = "mockServer";
031     private String dummyAnswer = "<var name=\"core.test.deep.struct.version\"
typ=\"d\" value=\"2.04\"/>\n" +
032         "<var name=\"rhd.linesensor\" typ=\"d\" size=\"9\" valid=\"true\"
value=\"1 53 51 53 53 54 54 54 54\"></var>\n" +
033         "<var name=\"rhd.irsensor\" typ=\"d\" size=\"7\" valid=\"true\" va
lue=\"1 169 107 90 89 2 255\"></var>";
034     private int port;
035
036     public MockServer(int port) {
037         this.port = port;
038         Thread t = new Thread(this);
039         t.start();
040     }
041
042     public void run() {
043         try {
044             String sentence;
045             ServerSocket serverSocket = new ServerSocket(port);
046             while (true) {
047                 System.out.println("Server Started: Listening on port " + port);
048                 Socket socket = serverSocket.accept();
049                 PrintWriter outToClient = new PrintWriter(socket.getOutputStream(), true);

```

```

050         BufferedReader inFromClient = new BufferedReader(new InputStre
amReader(socket.getInputStream()));
051         System.err.println("Connection OPENED. " + new Date());
052         System.out.println("Client IP: " + socket.getInetAddress());
053         outToClient.println("<" + rootElement + ">");
054         try {
055             while ((sentence = inFromClient.readLine()) != null) {
056                 System.out.println("Received cmd: " + sentence);
057                 lastReceivedCmd = sentence;
058                 outToClient.println(dummyAnswer);
059                 outToClient.flush();
060             }
061             outToClient.println("</"+ rootElement + ">");
062             System.err.println("Connection CLOSED.");
063             socket.close();
064         } catch (SocketException ex) {
065             System.err.println("Client DISCONNECTED");
066         }
067     }
068     } catch (IOException ex) {
069         Logger.getLogger(MockServer.class.getName()).log(Level.SEVERE, nul
l, ex);
070     }
071 }
072
073 public void setDummyAnswer(String dummyAnswer) {
074     this.dummyAnswer = dummyAnswer;
075 }
076
077 public String getLastReceivedCmd() {
078     return lastReceivedCmd;
079 }
080
081 public static void main(String[] args) {
082     MockServer mock = new MockServer(24928);
083 }
084
085 public void sendFileContents(File file, OutputStream out) {
086     try {
087         FileInputStream inFile = new FileInputStream(file);
088         copyContents(inFile, out);
089     } catch (FileNotFoundException ex) {
090         Logger.getLogger(MockServer.class.getName()).log(Level.SEVERE, nul
l, ex);
091     }
092 }
093
094 public void copyContents(InputStream from, OutputStream to) {
095     try {
096         byte[] buffer = new byte[512];
097         while ((from.read(buffer) != -1)) {
098             to.write(buffer);
099             buffer = new byte[512];
100         }
101         to.flush();
102     } catch (IOException ex) {
103         Logger.getLogger(MockServer.class.getName()).log(Level.SEVERE, nul
l, ex);
104     }
105 }
106 }

```

[illegible]

```

5));
059         outToClient.flush();
060         try {
061             Thread.sleep(1000);
062         } catch (InterruptedException ex) {
063             }
064         }
065     }
066     }
067     System.err.println("Connection CLOSED.");
068     socket.close();
069 }
070 } catch (IOException ex) {
071     Logger.getLogger(MockServerLineIR.class.getName()).log(Level.SEVERE,
E, null, ex);
072 }
073 }
074
075 private int getRandom(int min, int max) {
076     return min + (int) Math.round((Math.random() * (max - min)));
077 }
078
079 public String getLastReceivedCmd() {
080     return lastReceivedCmd;
081 }
082
083 public static void main(String[] args) {
084     MockServerLineIR mock = new MockServerLineIR(24929);
085 }
086
087 public void sendFileContents(File file, OutputStream out) {
088     try {
089         FileInputStream inFile = new FileInputStream(file);
090         copyContents(inFile, out);
091     } catch (FileNotFoundException ex) {
092     }
093 }
094 }
095
096 public void copyContents(InputStream from, OutputStream to) {
097     try {
098         byte[] buffer = new byte[512];
099         while ((from.read(buffer) != -1)) {
100             to.write(buffer);
101             buffer = new byte[512];
102         }
103         to.flush();
104     } catch (IOException ex) {
105         Logger.getLogger(MockServerLineIR.class.getName()).log(Level.SEVERE,
E, null, ex);
106     }
107 }
108 }

```

marg.test.mockserver.MockServerWheels

```
005 package marg.test.mockserver;
006
007 import java.io.BufferedReader;
008 import java.io.File;
009 import java.io.FileInputStream;
010 import java.io.FileNotFoundException;
011 import java.io.IOException;
012 import java.io.InputStream;
013 import java.io.InputStreamReader;
014 import java.io.OutputStream;
015 import java.io.PrintWriter;
016 import java.net.ServerSocket;
017 import java.net.Socket;
018 import java.net.SocketException;
019 import java.util.Date;
020 import java.util.logging.Level;
021 import java.util.logging.Logger;
022
023 /**
024  *
025  * @author MARG
026  */
027 public class MockServerWheels implements Runnable {
028
029     private String lastReceivedCmd = "";
030     private String rootElement = "mockServer";
031     private static final String LEFT_ENCODER_LINE = "<var name=\"rhd.encl\" ty
p=\"\" size=\"2\" value=\"1 %s\" desc=\"(r) first value is update flag\"/>";
032     private static final String RIGHT_ENCODER_LINE = "<var name=\"rhd.encl\" t
yp=\"\" size=\"2\" value=\"1 %s\" desc=\"(r) first value is update flag\"/>";
033     private static final String LEFT_SPEED_LINE = "<var name=\"rhd.speedl\" ty
p=\"d\" size=\"2\" value=\"1 %s\" desc=\"(w) writeable, first value unused\"/>";
034     private static final String RIGHT_SPEED_LINE = "<var name=\"rhd.speedr\" t
yp=\"d\" size=\"2\" value=\"1 %s\" desc=\"(w) writeable, first value unused\"/>";
035     private int encrLeft = 0;
036     private int encrRight = 0;
037     private int port;
038
039     public MockServerWheels(int port) {
040         this.port = port;
041         Thread t = new Thread(this);
042         t.start();
043     }
044
045     public void run() {
046         try {
047             String sentence;
048             ServerSocket serverSocket = new ServerSocket(port);
049             while (true) {
050                 System.out.println("Server Started: Listening on port " + port
);
051                 Socket socket = serverSocket.accept();
052                 PrintWriter outToClient = new PrintWriter(socket.getOutputStre
am(), true);
053                 BufferedReader inFromClient = new BufferedReader(new InputStre
amReader(socket.getInputStream()));
054                 System.err.println("Connection OPENED. " + new Date());
055                 System.out.println("Client IP: " + socket.getInetAddress());
056                 outToClient.println("<" + rootElement + ">");
057
058                 try {
059                     while ((sentence = inFromClient.readLine()) != null) {
```

```

060         System.out.println("Received cmd: " + sentence);
061         lastReceivedCmd = sentence;
062         for (int i = 0; i < 20; i++) {
063             encrLeft += 2000;
064             encrRight -= (int) (Math.random() * 600);
065             System.out.println("New values: " + encrLeft + " "
+ encrRight);
066             outToClient.println(String.format(LEFT_ENCODER_LIN
E, encrLeft));
067             outToClient.println(String.format(RIGHT_ENCODER_LI
NE, encrRight));
068             outToClient.println(String.format(RIGHT_SPEED_LINE
, -6));
069             outToClient.println(String.format(LEFT_SPEED_LINE,
20));
070             outToClient.flush();
071             try {
072                 Thread.sleep(1000);
073             } catch (InterruptedException ex) {
074                 Logger.getLogger(MockServerLineIR.class.getNam
e()).log(Level.SEVERE, null, ex);
075             }
076         }
077         outToClient.println(String.format(LEFT_ENCODER_LINE, e
ncrLeft));
078         outToClient.println(String.format(RIGHT_ENCODER_LINE,
encrRight));
079     }
080     outToClient.println("</" + rootElement + ">");
081     System.err.println("Connection CLOSED.");
082     socket.close();
083     } catch (SocketException ex) {
084         System.err.println("Client DISCONNECTED");
085     }
086 }
087 } catch (IOException ex) {
088     Logger.getLogger(MockServerWheels.class.getName()).log(Level.SEVERE,
null, ex);
089 }
090 }
091
092 public String getLastReceivedCmd() {
093     return lastReceivedCmd;
094 }
095
096 public static void main(String[] args) {
097     MockServerWheels mock = new MockServerWheels(24928);
098 }
099
100 public void sendFileContents(File file, OutputStream out) {
101     try {
102         FileInputStream inFile = new FileInputStream(file);
103         copyContents(inFile, out);
104     } catch (FileNotFoundException ex) {
105         Logger.getLogger(MockServerWheels.class.getName()).log(Level.SEVERE,
null, ex);
106     }
107 }
108
109 public void copyContents(InputStream from, OutputStream to) {
110     try {
111         byte[] buffer = new byte[512];
112         while ((from.read(buffer) != -1)) {
113             to.write(buffer);

```

```

114         buffer = new byte[512];
115     }
116     to.flush();
117 } catch (IOException ex) {
118     Logger.getLogger(MockServerWheels.class.getName()).log(Level.SEVERE
E, null, ex);
119 }
120 }
121 }

```

marg.util.Log

```

06 package marg.util;
07
08 import java.util.logging.Logger;
09
10 /**
11  *
12  * @author MARG
13  */
14 public class Log {
15
16     public static final Logger GlobalLogger = Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);
17
18 }

```

marg.util.PluginManager

```
005 package marg.util;
006
007 import java.io.File;
008 import java.io.IOException;
009 import java.io.UnsupportedEncodingException;
010 import java.net.JarURLConnection;
011 import java.net.URL;
012 import java.net.URLDecoder;
013 import java.util.ArrayList;
014 import java.util.Arrays;
015 import java.util.Collections;
016 import java.util.Enumeration;
017 import java.util.List;
018 import java.util.jar.JarEntry;
019 import java.util.jar.JarFile;
020 import java.util.logging.Level;
021 import java.util.logging.Logger;
022 import marg.gui.plugin.ModulePlugin;
023 import marg.model.plugin.XMLParsePlugin;
024
025 /**
026  * Developed with help from http://forums.sun.com/thread.jspa?threadID=341935&start=15 to get classes from a package inside a loaded jar
027  * @author MARG
028  */
029 public class PluginManager {
030
031     private static PluginManager instance;
032     List<XMLParsePlugin> parsePlugins = new ArrayList<XMLParsePlugin>();
033     List<Class> modulePluginClasses = new ArrayList<Class>();
034
035     public static PluginManager getInstance() {
036         if (instance == null)
037             instance = new PluginManager();
038         return instance;
039     }
040
041     private PluginManager() {
042         //initParsePlugins(); TODO: Enable auto-detection of parseplugins
043         initModulePlugins();
044     }
045
046     private void initModulePlugins() {
047         modulePluginClasses = getClassessOfInterface("marg.gui.plugin", Module
Plugin.class);
048     }
049
050     private void initParsePlugins() {
051         List<Class> pluginClasses = getClassessOfInterface("marg.model.plugin"
, XMLParsePlugin.class);
052         createClassesAndFillIntoList(pluginClasses, parsePlugins);
053     }
054
055     private <T> void createClassesAndFillIntoList(List<Class> classes, List<T>
container) {
056         List<String> addedPlugins = new ArrayList<String>(); //Keeping track s
o we do not add the same plugin more than once.
057         for (Class aClass : classes) {
058             if (!addedPlugins.contains(aClass.getSimpleName())) {
059                 //System.out.println("Creating class: " + aClass);
060                 Object inst = getInstanceOfClass(aClass); //Refactored to get
rid of try-catch clutter

```



```

061         if (inst != null) {
062             container.add((T) inst);
063             addedPlugins.add(aClass.getSimpleName());
064         }
065     }
066 }
067
068
069 public List<XMLParsePlugin> getParsePlugins() {
070     return parsePlugins;
071 }
072
073 public List<Class> getAvailableModulePlugins() {
074     return modulePluginClasses;
075 }
076
077 public Object getInstanceOfClass(Class aClass) {
078     Object returnObject = null;
079     try {
080         returnObject = aClass.newInstance();
081     } catch (InstantiationException ex) {
082         Logger.getLogger(PluginManager.class.getName()).log(Level.SEVERE,
083 null, ex);
084     } catch (IllegalAccessException ex) {
085         Logger.getLogger(PluginManager.class.getName()).log(Level.SEVERE,
086 null, ex);
087     }
088     return returnObject;
089 }
090
091 private static List<Class> getClassesForPackage(String pckgname)
092     throws ClassNotFoundException {
093     // This will hold a list of directories matching the pckgname.
094     // There may be more than one if a package is split over multiple jars/
095     paths
096     List<Class> classes = new ArrayList<Class>();
097     ArrayList<File> directories = new ArrayList<File>();
098     try {
099         ClassLoader cld = Thread.currentThread().getContextClassLoader();
100         if (cld == null) {
101             throw new ClassNotFoundException("Can't get class loader.");
102         }
103         // Ask for all resources for the path
104         Enumeration<URL> resources = cld.getResources(pckgname.replace('.',
105 '/', '/'));
106         while (resources.hasMoreElements()) {
107             URL res = resources.nextElement();
108             if (res.getProtocol().equalsIgnoreCase("jar")) {
109                 JarURLConnection conn = (JarURLConnection) res.openConnection();
110                 JarFile jar = conn.getJarFile();
111                 for (JarEntry e : Collections.list(jar.entries())) {
112                     if (e.getName().startsWith(pckgname.replace('.', '/'))
113 && e.getName().endsWith(".class") && !e.getName().contains("$")) {
114                         String className =
115                             e.getName().replace("/", ".").substring(0,
116 e.getName().length() - 6);
117                         classes.add(Class.forName(className));
118                     }
119                 }
120             } else {
121                 directories.add(new File(URLDecoder.decode(res.getPath(),
122 "UTF-8")));
123             }
124         }
125     }
126 }

```

```

117     }
118     } catch (NullPointerException x) {
119         throw new ClassNotFoundException(pckgname + " does not appear to b
120         e " +
121         "a valid package (Null pointer exception)");
122     } catch (UnsupportedEncodingException encex) {
123         throw new ClassNotFoundException(pckgname + " does not appear to b
124         e " +
125         "a valid package (Unsupported encoding)");
126     } catch (IOException ioex) {
127         throw new ClassNotFoundException("IOException was thrown when tryi
128         ng " +
129         "to get all resources for " + pckgname);
130     }
131
132     // For every directory identified capture all the .class files
133     for (File directory : directories) {
134         if (directory.exists()) {
135             // Get the list of the files contained in the package
136             String[] files = directory.list();
137             for (String file : files) {
138                 // we are only interested in .class files
139                 if (file.endsWith(".class")) {
140                     // removes the .class extension
141                     classes.add(Class.forName(pckgname + '.' + file.substr
142                     ing(0, file.length() - 6)));
143                 }
144             } else {
145                 throw new ClassNotFoundException(pckgname + " (" + directory.g
146                 etPath() +
147                 ") does not appear to be a valid package");
148             }
149         }
150         return classes;
151     }
152 }
153
154 private static List<Class> getClassessOfInterface(String thePackage, Class
theInterface) {
155     List<Class> classList = new ArrayList<Class>();
156     try {
157         for (Class discovered : getClassesForPackage(thePackage)) {
158             if (Arrays.asList(discovered.getInterfaces()).contains(theInte
159             rface)) {
160                 if (!classList.contains(discovered)) //We don't want dupli
161                 cates of same implementation.
162                 classList.add(discovered);
163             }
164         }
165     } catch (ClassNotFoundException ex) {
166         Logger.getLogger(PluginManager.class.getName()).log(Level.SEVERE,
167         null, ex);
168     }
169     return classList;
170 }
171
172 public static void main(String[] args) {
173     for (Class aClass : PluginManager.getInstance().getAvailableModulePlug
174     ins()) {
175         System.out.println("Found Plugin: "+ aClass.getName());
176     }
177 }
178 }
179 }

```

marg.XMLParsePluginTemplate

```
06 package marg;
07
08 import marg.model.plugin.*;
09 import java.beans.PropertyChangeSupport;
10 import org.xml.sax.Attributes;
11
12 /**
13  *
14  * @author MARG
15  */
16 public class XMLParsePluginTemplate implements XMLParsePlugin {
17
18     private PropertyChangeSupport prop;
19
20     public void setPropertyChangeSupport(PropertyChangeSupport prop) {
21         this.prop = prop;
22     }
23
24     /**
25      * Called when the starttag of an element is received
26      * @param tagName name of the element/tag
27      * @param atts attributes of the element
28      */
29     public void startElement(String tagName, Attributes atts) {
30         //Example: Firing a callback
31         //All listeners who has subscribed to "propertyName" will receive this
callback
32         prop.firePropertyChange("propertyName", "oldValue", tagName);
33         //End of Example
34     }
35
36     /**
37      * Called when the endtag of an element is received
38      * @param tagName name of the element/tag
39      */
40     public void endElement(String tagName) {
41         throw new UnsupportedOperationException("Not supported yet.");
42     }
43
44     /**
45      * Called when contents of an element is received. Data can be of any kind
46      * @param parentTag name of the first parent tag this content was wrapped i
n
47      * @param ch character buffer
48      * @param start start offset for the content in character buffer
49      * @param length length of content
50      */
51     public void tagContents(String parentTag, char[] ch, int start, int length)
{
52         throw new UnsupportedOperationException("Not supported yet.");
53     }
54
55 }
```

marg.ModulePluginTemplate

```
01 package marg;
02
03 import marg.gui.plugin.*;
04 import javax.swing.JPanel;
05 import marg.handlers.XMLClientHandler;
06
07 /**
08  * Insert description of this plugin
09  * @author Insert author of plugin
10  */
11 public class ModulePluginTemplate implements ModulePlugin {
12
13     private XMLClientHandler handler;
14
15     public void setXMLClientHandler(XMLClientHandler handler) {
16         this.handler = handler;
17     }
18
19     /**
20      * Gets the name of your plugin.
21      * Used as the header for the tab of your plugin.
22      * @return a String representing the name of your plugin
23      */
24     public String getPluginName() {
25         throw new UnsupportedOperationException("Not supported yet.");
26     }
27
28     /**
29      * Return the visual representation of your plugin here.
30      * Parent module calls this method to retrieve a JPanel and adds it to its
31      * tabbed pane.
32      * @return a JPanel object created by your plugin
33      */
34     public JPanel getJPanel() {
35         throw new UnsupportedOperationException("Not supported yet.");
36     }
37
38     /**
39      * Called by the parent module to start a plugin.
40      * Use this method to initialize ressources that are not needed until
41      * your plugin is actually started
42      */
43     public void startPlugin() {
44         throw new UnsupportedOperationException("Not supported yet.");
45     }
46
47     /**
48      * Called by the parent module during cleanup process when the module or en
49      * tire applet is closed.
50      * Clean-up of ressources should be placed here.
51      */
52     public void stopPlugin() {
53         throw new UnsupportedOperationException("Not supported yet.");
54     }
55
56     /**
57      * Called by parent module regularly every second by default.
58      * Saves you having to make a Thread if you need a simple regular update in
59      * your plugin.
60      */
61     public void doUpdate() {
62         throw new UnsupportedOperationException("Not supported yet.");
63     }
64 }
```

```

60     }
61
62 }

```

marg.ModulePluginJPanelTemplate

```

011 package marg;
012
013 import marg.gui.plugin.*;
014 import java.beans.PropertyChangeEvent;
015 import java.beans.PropertyChangeListener;
016 import javax.swing.JPanel;
017 import marg.handlers.XMLClientHandler;
018 import marg.model.ModuleVariable;
019 import marg.model.plugin.VarDataPlugin;
020
021 /**
022  *
023  * @author MARG
024  */
025 public class ModulePluginJPanelTemplate extends javax.swing.JPanel implements
ModulePlugin, PropertyChangeListener {
026
027     private XMLClientHandler clientHandler;
028
029     /** Creates new form ModulePluginJPanelTemplate */
030     public ModulePluginJPanelTemplate() {
031         initComponents();
032     }
033
034     /** This method is called from within the constructor to
035      * initialize the form.
036      * WARNING: Do NOT modify this code. The content of this method is
037      * always regenerated by the Form Editor.
038      */
039     @SuppressWarnings("unchecked")
040     // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
BEGIN:initComponents
041     private void initComponents() {
063     } // Removed auto-generated code
064     // Variables declaration - do not modify //GEN-BEGIN:variables
065     private javax.swing.JLabel jLabel1;
066     // End of variables declaration //GEN-END:variables
067
068     //This will always be called before startPlugin()
069     public void setXMLClientHandler(XMLClientHandler handler) {
070         this.clientHandler = handler;
071     }
072
073     public String getPluginName() {
074         return "Template";
075     }
076
077     public JPanel getJPanel() {
078         return this;
079     }
080
081     public void startPlugin() {
082         //Initialize needed ressources
083         //Handler usage example:
084         clientHandler.addPropertyChangeListener(this, VarDataPlugin.SUBSCRIB
E_VARDATA);
085     }
086
087     public void stopPlugin() {

```

```

088         //Close down ressources
089     }
090
091     public void doUpdate() {
092         //Update some content
093     }
094
095     public void propertyChange(PropertyChangeEvent evt) {
096         if (evt.getPropertyName().equals(VarDataPlugin.SUBSCRIBE_VARDATA)) {
097             System.out.println("Plugin received a variable!");
098             ModuleVariable variable = (ModuleVariable) evt.getNewValue();
099             System.out.println("Variable name: " + variable.getVarName());
100             System.out.println("Variable value: " + variable.getValue());
101         }
102     }
103 }

```

JUnit MARGTestSuite

```

06 package marg.model;
07
08 import org.junit.After;
09 import org.junit.AfterClass;
10 import org.junit.Before;
11 import org.junit.BeforeClass;
12 import org.junit.runner.RunWith;
13 import org.junit.runners.Suite;
14
15 /**
16  *
17  * @author MARG
18  */
19 @RunWith(Suite.class)
20 @Suite.SuiteClasses({marg.model.XMLClientTest.class,marg.model.MonitoredVariableTest.class,marg.model.ValueLimitTest.class})
21 public class MARGTestSuite {
22
23     @BeforeClass
24     public static void setUpClass() throws Exception {
25     }
26
27     @AfterClass
28     public static void tearDownClass() throws Exception {
29     }
30
31     @Before
32     public void setUp() throws Exception {
33     }
34
35     @After
36     public void tearDown() throws Exception {
37     }
38
39 }

```

JUnit MonitoredVariableTest

```
06 package marg.model;
07
08 import org.junit.After;
09 import org.junit.AfterClass;
10 import org.junit.Before;
11 import org.junit.BeforeClass;
12 import org.junit.Test;
13 import static org.junit.Assert.*;
14
15 /**
16  *
17  * @author MARG
18  */
19 public class MonitoredVariableTest {
20
21     private ModuleVariable modVar;
22     private MonitoredVariable monVar;
23     private String presentation = "";
24
25     public MonitoredVariableTest() {
26     }
27
28     @BeforeClass
29     public static void setUpClass() throws Exception {
30     }
31
32     @AfterClass
33     public static void tearDownClass() throws Exception {
34     }
35
36     @Before
37     public void setUp() {
38         presentation = "Speedr: %s cm/s";
39         modVar = new ModuleVariable("rhd.speedr", "d", "20");
40         monVar = new MonitoredVariable(modVar, presentation);
41     }
42
43     @After
44     public void tearDown() {
45     }
46
47     /**
48      * Test of getPresentation method, of class MonitoredVariable.
49      */
50     @Test
51     public void testGetPresentation() {
52         System.out.println("getPresentation");
53         String expResult = "Speedr: %s cm/s";
54         assertEquals(expResult, monVar.getPresentation());
55     }
56
57     /**
58      * Test of getModuleVar method, of class MonitoredVariable.
59      */
60     @Test
61     public void testGetModuleVar() {
62         System.out.println("getModuleVar");
63         ModuleVariable expResult = modVar;
64         ModuleVariable result = monVar.getModuleVar();
65         assertEquals(expResult, result);
66     }
67 }
```

```

68  /**
69   * Test of setPresentation method, of class MonitoredVariable.
70   */
71  @Test
72  public void testSetPresentation() {
73      System.out.println("setPresentation");
74      String newPresentation = "Some value: %s";
75      monVar.setPresentation(newPresentation);
76      assertEquals(newPresentation, monVar.getPresentation());
77  }
78
79  /**
80   * Test of toString method, of class MonitoredVariable.
81   */
82  @Test
83  public void testToString() {
84      System.out.println("toString");
85      String expResult = presentation.replace("%s", modVar.getValue());
86      String result = monVar.toString();
87      assertEquals(expResult, result);
88  }
89
90 }

```


JUnit ValueLimitTest

```
006 package marg.model;
007
008 import org.junit.After;
009 import org.junit.AfterClass;
010 import org.junit.Before;
011 import org.junit.BeforeClass;
012 import org.junit.Test;
013 import static org.junit.Assert.*;
014
015 /**
016  *
017  * @author MARG
018  */
019 public class ValueLimitTest {
020
021     private ValueLimit limitLEQ;
022     private ValueLimit limitL;
023
024     public ValueLimitTest() {
025     }
026
027     @BeforeClass
028     public static void setUpClass() throws Exception {
029     }
030
031     @AfterClass
032     public static void tearDownClass() throws Exception {
033     }
034
035     @Before
036     public void setUp() {
037         limitLEQ = new ValueLimit(-
038 50.5, ValueLimit.LESS_THAN_EQUALS, 50.5, ValueLimit.LESS_THAN_EQUALS);
039         limitL = new ValueLimit(-
040 50.5, ValueLimit.LESS_THAN, 50.5, ValueLimit.LESS_THAN);
041     }
042
043     @After
044     public void tearDown() {
045     }
046
047     /**
048      * Test of checkCorrect method, of class ValueLimit.
049      */
050     @Test
051     public void testCheckCorrect() {
052         System.out.println("checkCorrect");
053         double value = -50.5;
054         assertEquals(true, limitLEQ.checkCorrect(value));
055         assertEquals(false, limitL.checkCorrect(value));
056
057         value = 0;
058         assertEquals(true, limitLEQ.checkCorrect(value));
059         assertEquals(true, limitL.checkCorrect(value));
060
061         value = 50.5;
062         assertEquals(true, limitLEQ.checkCorrect(value));
063         assertEquals(false, limitL.checkCorrect(value));
064     }
065 }
```

```

064  /**
065   * Test of getMaxValue method, of class ValueLimit.
066   */
067  @Test
068  public void testGetMaxValue() {
069      System.out.println("getMaxValue");
070      double expResult = 50.5;
071      double result = limitLEQ.getMaxValue();
072      assertEquals(expResult, result, 0.0);
073  }
074
075  /**
076   * Test of getMinValue method, of class ValueLimit.
077   */
078  @Test
079  public void testGetMinValue() {
080      System.out.println("getMinValue");
081      double expResult = -50.5;
082      double result = limitLEQ.getMinValue();
083      assertEquals(expResult, result, 0.0);
084
085      result = limitL.getMinValue();
086      assertEquals(expResult, result, 0.0);
087  }
088
089  /**
090   * Test of getMaximumComparison method, of class ValueLimit.
091   */
092  @Test
093  public void testGetMaximumComparison() {
094      System.out.println("getMaximumComparison");
095      assertEquals(ValueLimit.LESS_THAN, limitL.getMaximumComparison());
096      assertEquals(ValueLimit.LESS_THAN_EQUALS, limitLEQ.getMaximumCompariso
n());
097  }
098
099  /**
100   * Test of getMinimumComparison method, of class ValueLimit.
101   */
102  @Test
103  public void testGetMinimumComparison() {
104      System.out.println("getMinimumComparison");
105      assertEquals(ValueLimit.LESS_THAN, limitL.getMinimumComparison());
106      assertEquals(ValueLimit.LESS_THAN_EQUALS, limitLEQ.getMinimumCompariso
n());
107  }
108
109  /**
110   * Test of toString method, of class ValueLimit.
111   */
112  @Test
113  public void testToString() {
114      System.out.println("toString");
115      String expResult = "ValueLimit (min -
50.5, max 50.5, MinLTE false, MaxLTE false)";
116      assertEquals(expResult, limitL.toString());
117
118      expResult = "ValueLimit (min -
50.5, max 50.5, MinLTE true, MaxLTE true)";
119      assertEquals(expResult, limitLEQ.toString());
120  }
121
122  }

```

JUnit XMLClientTest

```
005 package marg.model;
006
007 import marg.test.mockserver.JUnitMockServer;
008 import org.junit.After;
009 import org.junit.AfterClass;
010 import org.junit.Before;
011 import org.junit.BeforeClass;
012 import org.junit.Test;
013 import static org.junit.Assert.*;
014
015 /**
016  *
017  * @author MARG
018  */
019 public class XMLClientTest {
020
021     public XMLClientTest() {
022     }
023
024     @Before
025     public void setUp() {
026     }
027
028     @After
029     public void tearDown() {
030     }
031
032     /**
033      * Test of sendCmd method, of class XMLClient.
034      */
035     @Test
036     public void testSendCmd() throws InterruptedException {
037         System.out.println("sendCmd");
038         JUnitMockServer mock = new JUnitMockServer(24001);
039         XMLClient client = new XMLClient();
040         client.connect("localhost", 24001);
041         client.sendCmd("testing");
042         String expResult = "<testing/>";
043         Thread.sleep(200); //Delay for the server thread
044         String result = mock.getLastReceivedCmd();
045         assertEquals(expResult, result);
046     }
047
048     /**
049      * Test of getXMLParser method, of class XMLClient.
050      */
051     @Test
052     public void testGetXMLParser() {
053         System.out.println("getXMLParser");
054         XMLClient instance = new XMLClient();
055         XMLParser result = instance.getXMLParser();
056         assertTrue(result instanceof XMLParser);
057     }
058
059     @Test
060     public void testConnect() {
061         XMLClient client = new XMLClient();
062         try {
063             client.connect("localhost", 65536); //Port out of bound
064             fail("Expected IllegalArgumentException");
065         }
066     }
067 }
```

```

073         } catch (IllegalArgumentException ex) {}
074     try {
075         client.connect("localhost", 1023); //Port out of bound
076         fail("Expected IllegalArgumentException");
077     } catch (IllegalArgumentException ex) {}
078     assertFalse(client.isConnected());
079 }
080
081 @Test
082 public void testConnect2() {
083     JUnitMockServer mock = new JUnitMockServer(65535); //Highes possible p
084     XMLClient client = new XMLClient();
085     client.connect("localhost", 65535);
086     assertEquals(client.isConnected(), true);
087 }
088
089 @Test
090 public void testConnect3() {
091     JUnitMockServer mock = new JUnitMockServer(1024); //Lowest possible po
092     XMLClient client = new XMLClient();
093     client.connect("localhost", 1024);
094     assertEquals(client.isConnected(), true);
095 }
096
097 @Test
098 public void testDisconnect() {
099     JUnitMockServer mock = new JUnitMockServer(24002);
100     XMLClient client = new XMLClient();
101     client.connect("localhost", 24002);
102     assertTrue(client.isConnected());
103     client.disconnect();
104     assertFalse(client.isConnected());
105 }
106 }

```