Arkitekturdokument

Patrik Lundgren

Version 1.0

Status

Granskad	Henrik	2017-02-17
Godkänd		

PROJEKTIDENTITET

VT17, Grupp 6 - Point of Interest
Linköpings tekniska högskola, IDA

Namn	Ansvar	Telefon	E-post
Daniel Persson Proos	Utvecklingsledare	070-091 15 56	danpr535@student.liu.se
Fredrik Iselius	Testledare	070-695 61 02	freis685@student.liu.se
George Yildiz	Teamledare	076-049 57 15	geoyi478@student.liu.se
Henrik Persson	Dokument- och konfigurationsansvarig	073-426 52 64	henpe071@student.liu.se
Kristian Nilsson	Analysansvarig	070-936 97 63	krini678@student.liu.se
Patrik Lundgren	Arkitekt	073-642 72 54	patlu721@student.liu.se
Pär Sörliden	Kvalitetssamordnare	076-595 59 50	parso619@student.liu.se

Kund: Nationellt forensiskt centrum (NFC), Brigadgatan 13, Garnisonen, 587 58 LINKÖPING,
kundtelefon 010-562 80 00, fax: 013-14 57 15, registrator.nfc@polisen.se
Kontaktperson hos kund: Niclas Appleby, niclas.appleby@polisen.se, 010-562 84 58
Erik Öhrn, erik.ohrn@polisen.se

Kursansvarig: Kristian Sandahl, B 3B:470, 013-28 19 57, kristian.sandahl@liu.se
Handledare: Kimberley French, kimberley.french@liu.se

Innehåll

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
1.0	2017-02-17	Första versionen	Henrik	
0.1	2017-02-16	Första utkastet		Henrik

1 INLEDNING

Detta dokument beskriver den övergripande designfilosofin som använts i projektet, begränsningar på systemet, beslut som tagits och varför de tagits. Vidare beskriver detta dokument även de designmönster som ska användas, var de ska användas samt viktiga abstraktioner. Dokumentet är skrivet enligt en mall från openUP [1].

1.1 Definitioner

Nedan följer definitioner som kommer att användas i delar av dokumentet.

POI	(Point of interest) Video- eller bildintervall som analysverktyget markerat.
OOI	(Object of interest) Objekt i bild som är av intresse.
MVC	(Model-View-Controller) Designmönster, se rubrik 3.2.

1.2 Mål & filosofi

Målet med designen av programmet är att skapa ett robust och funktionellt system som underlättar vidareutveckling. Drivande för denna fråga är att vidareutveckling av analysverktyg skall vara så enkelt som möjligt. Utifrån detta har ett antal designmål tagits fram.

1. Programmet ska erbjuda ett tydligt och flexibelt gränssnitt mot analysmodulen för underlättad utökning av denna.
2. Det grafiska gränssnittets funktionalitet ska vara lättåtkomligt för att underlätta vidare implementation.
3. Ändringar i programmet i syfte att implementera ett nytt analysverktyg ska vara så enhetliga som möjligt för att underlätta felsökning och implementation.

1.3 Riktlinjer

Under utvecklingen av produkten ges ett antal riktlinjer för förändringar i arkitekturen av programvaran. Detta är till viss del i dokumentationssyfte men också i syfte att säkerställa en enhetlig produkt med väl kompatibla komponenter under utveckling.

1. Gränssnitt ska vara väl dokumenterade innan de erbjuds i produkten.
2. Vid icke-triviala designbeslut ska filosofin, målen och kraven i detta dokument beaktas. Förslaget bör även diskuteras med arkitekten.
3. Designbeslut som påverkar gränssnitt ska diskuteras med arkitekten och påverkade utvecklare innan implementation.
4. Förändringar av redan beslutad arkitektur ska vara motiverade och diskuteras.

.

1.4 Antaganden och beroenden

Designlösningarna begränsas i huvudsak av att analysmodulen ska använda sig av biblioteket OpenCV men också av att GUI:t ska vara baserat på grafikbiblioteket QT. Dessutom ska hela utvecklingen ske i C++.

1.5 Viktiga krav

Nedanstående krav har bedömts viktiga för arkitekturen av olika anledningar. Dessa krav i synnerhet bör finnas i åtanke vid designbeslut som påverkar gränssnitt och arkitektur. De är hämtade från kravspecifikationen [2].

Nr	Beskrivning	Varför är det viktigt?
16	Användaren ska kunna specificera ett område i filmen där analysen ska köras.	Detta krav sätter potentiellt begränsningar på hur analysverktygen ska fungera.
18	Användaren ska kunna köa flera filmer/bilder för analys.	Krav på flexibla analysverktyg.
43	Om fler än ett videoklipp spelas ska man kunna styra dem enskilt och tillsammans.	Detta krav har ett behov av en flexibel videospelare och ger krav på att användaren kan välja vilken video som ska kontrolleras.
59	Det ska gå att lägga till fler typer av analyser i efterhand.	Detta krav har ett indirekt behov av tydliga gränssnitt mot analysverktyget och låg sammankoppling, detta för att underlätta vidareutveckling.
60	När en analys körs ska det vara möjligt att använda andra funktioner i programmet.	Detta resulterar i att GUI:t måste köra på en egen tråd, samt att videospelaren måste köra på en egen tråd.

2 DESIGNBESLUT, BEGRÄNSNING OCH MOTIVERING

Nedan beskrivs några designbeslut och begränsningar inom projektet.

2.1 MVC

Beslutet att arbeta efter MVC (Model-View-Controller) är för att separera funktionalitet från uppvisning, se rubrik 3.2. Detta beslut togs för att det ska underlätta parallell utveckling av användargränssnitt och funktionalitet.

2.2 GUI - Kommunikation via kö

Det har beslutats att kommunikation mellan gränssnittet och resterande moduler ska ske genom en kö. I denna kö läggs förfrågningar på gränssnittsvisningar, dessa förfrågningar behandlas regelbundet av GUI:t för att hålla allt korrekt uppdaterat.

2.3 Videospelare - Observatörmönster

Det har beslutats att filmer i videospelaren som spelas och kontrolleras samtidigt ska följa ett observatörmönster, se rubrik 3.1. Detta beslut togs för att möjliggöra ett abstraktionslager för kontroll av flera videor och för att göra denna implementation så minimal som möjlig.

2.4 Analys - Analyskö

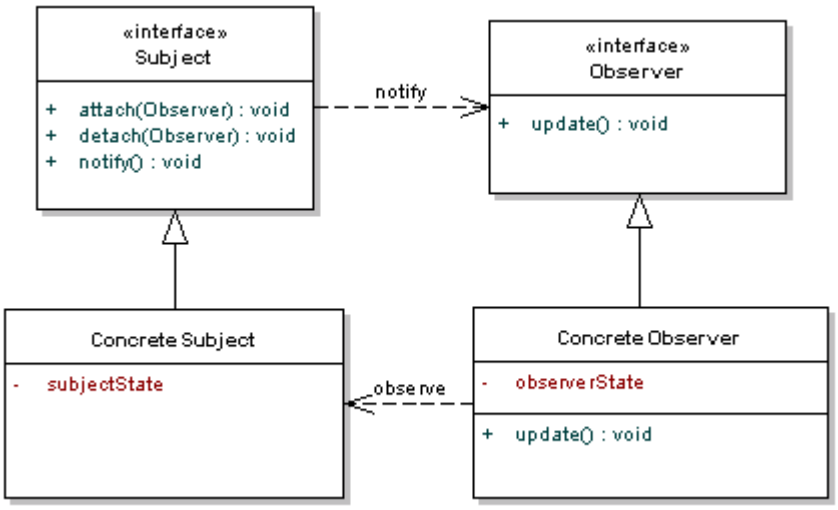
Det har beslutats att det endast ska gå att köra en analys i taget och att ytterligare begäran på analyser ska resultera i att de läggs på kö. Detta beslut togs för att begränsa komplexiteten i kommunikationen mellan trådar.

3 DESIGNMÖNSTER

Här beskrivs några designmönster som kommer att användas.

3.1 Observatörmönstret.

Observatör är ett mönster som lämpar sig för interaktiva system där ett antal delar av programmet är intresserade av samma händelse [3]. De delar som är intresserade av en händelse läggs då till som observatör av de subjekt som vet när händelsen infaller. Detta subjekt notifierar då alla objekt vid händelsen.



Figur 1: Diagram över observatörmönstret [6].

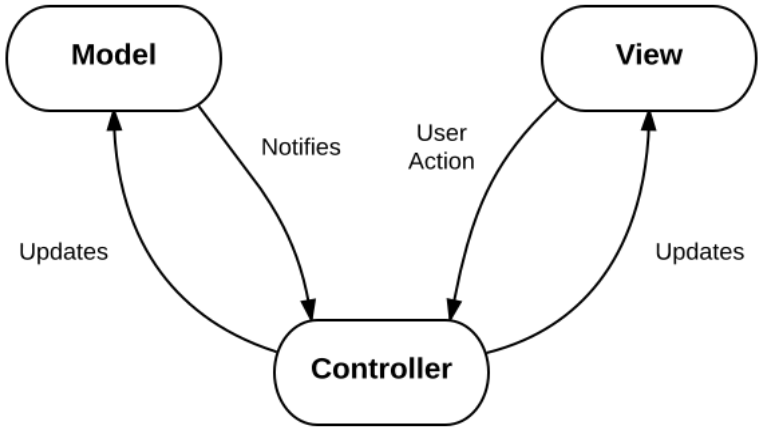
3.1.1 Videospelare

Observer lämpar sig väl för VideoController i videospelaren eftersom denna ska kontrollera uppspelningen av ett flertal videor. En lämplig arbetsgång blir då att implementera inställningar för videoobjekten (såsom spelhastighet, tidpunkt o.s.v.) och sedan notifiera dessa objekt beroende på användarens inmatning.

3.2 Model-View-Controller

Syftet med Model-View-Controller är att separera de logiskt skilda delarna i ett gränssnitt. Datamanipulation ska separeras från uppvisning för att göra visning av data flexibel och oberoende av datamanipulation. Vidare så ska användarinmatning vara separerad från programmets funktionalitet, detta ökar programmets testbarhet eftersom Model kan testas separat från användarinmatning [4].

- Model är den centrala komponenten i mönstret. Det är denna komponent som behandlar den funktionalitet som inte direkt beror av användarinmatning. Logik och funktionalitet som används internt ska finnas här. Även användarinställningar ska finnas här.
- View är visningskomponenten. Den kan betraktas som en projicering av Model-komponenten. View ska inte ändra data i programmet, den ska endast ändras beroende på Model och de användarinställningar som finns.
- Controller är inmatningskomponenten. I denna komponent ska inmatning från användaren registreras för att manipulera Model.



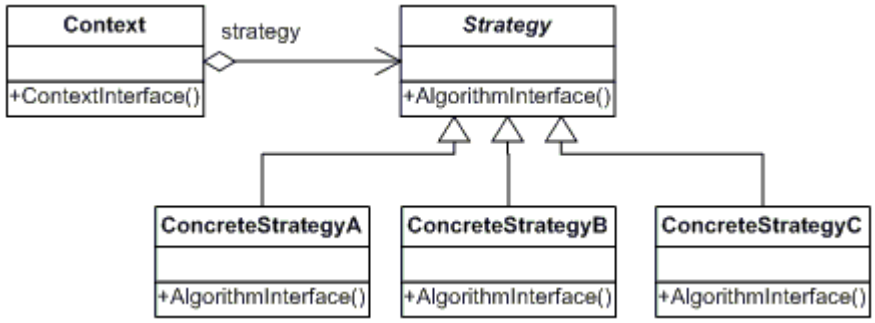
Figur 2: Diagram som visar strukturen i MVC [4].

3.2.1 GUI

MVC-mönstret lämpar sig för GUI:t. Detta för att separera intern datamanipulation, användarinmatning och vy.

3.3 Strategimönster

Strategimönstret är ett implementationsmönster som går ut på att identifiera en familj av algoritmer som kan erbjuda och svara på ett likadant gränssnitt [5]. Ett enhetligt gränssnitt kan då avsevärt förenkla hanteringen av denna familj av algoritmer.



Figur 3: UML-diagram för strategimönstret [7].

3.3.1 Analysverktyg

Strategimönstret lämpar sig för analysverktyg eftersom detta skulle göra gränssnittet mot resten av programmet mer enhetligt. Det skulle även potentiellt underlätta utveckling av nya analysmetoder, då dessa endast behöver uppfylla gränssnittet för strategimönstret.

4 VIKTIGA ABSTRAKTIONER

Detta avsnitt beskriver viktiga abstraktioner som påverkar arkitekturen nämnvärt. Dessa abstraktioner behövs för att göra arkitekturen stabil.

4.1 GUI

Det är viktigt att gränssnittet mot GUI-modulen för att skapa grafiska objekt är abstraherad från QT-biblioteket. Detta är för att man vid vidareutveckling ska kunna göra anpassningar av gränssnittet utan utförlig kunskap om QT-biblioteket.

4.2 Analysverktyg

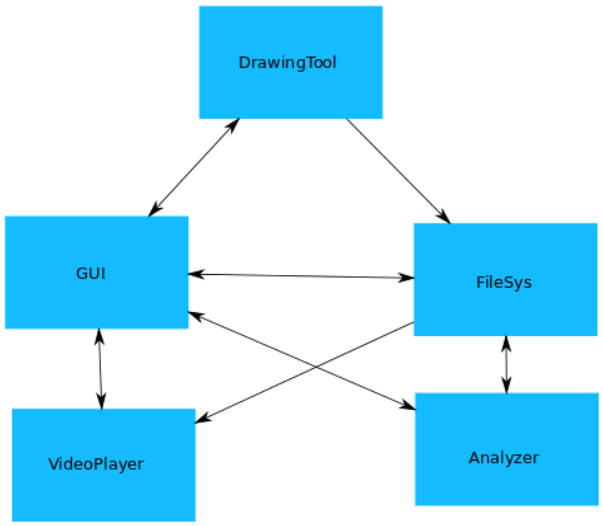
Det är viktigt att analysverktygen abstraheras till en generell form. Detta för att implementation av en ny analysmetod inte ska resultera i stora sidoeffekter i resten av programmet.

5 SYSTEMET

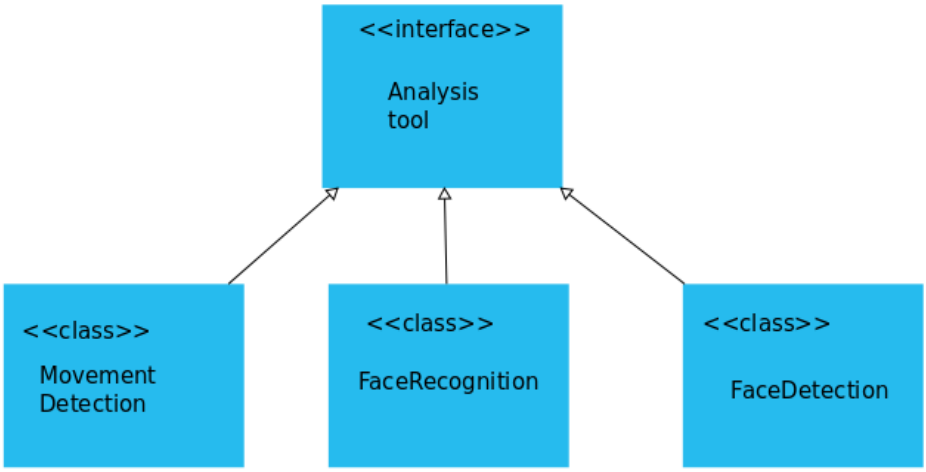
Systemet kommer att bestå av en videospelare. Ett lager som man kan rita på för att markera ut saker på bilden kommer att finnas ovanpå videon. Ett filhanteringssystem kommer att finnas för att ladda in videoklipp och spara undan bilder till en dokumentationslista.

5.1 Kommunikation

Kommunikationen i programmet kommer i synnerhet att ske direkt med användarna genom det grafiska gränssnittet samt genom filhanteringssystemet, se Figur 4.



Figur 5: Överblick over programmets kommunikation.



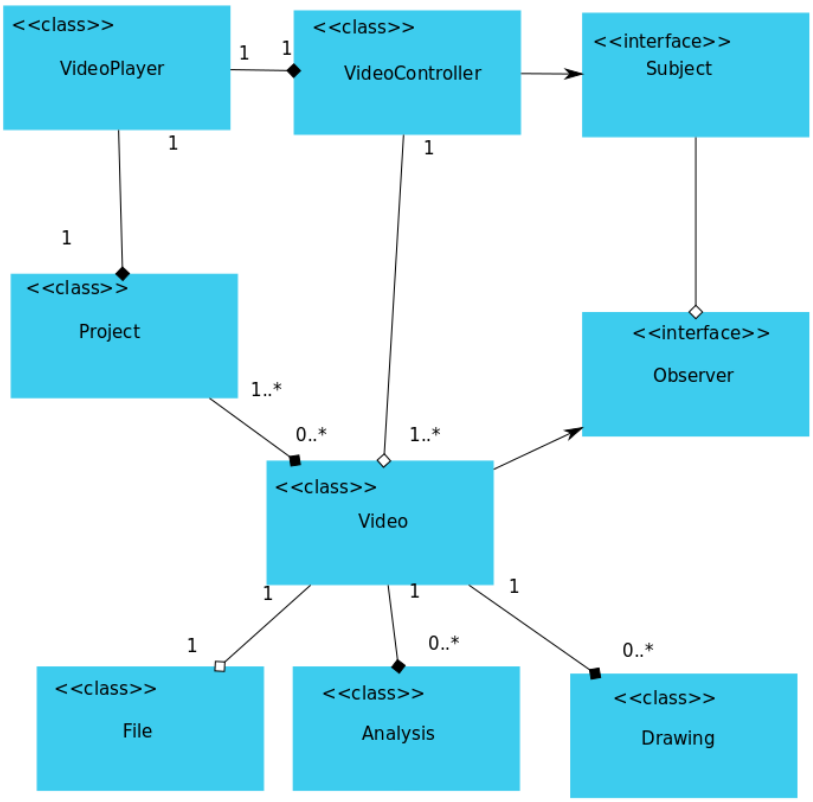
Figur 4: Illustration av strategimönster i analysverktyget.

5.2 Analysverktyg

Analysverktyget har strukturerats på följande sätt med ett strategimönster, se rubrik 3.3. Detta för att underlätta implementation vid tillägg av ytterligare analysverktyg.

5.3 Videospelaren

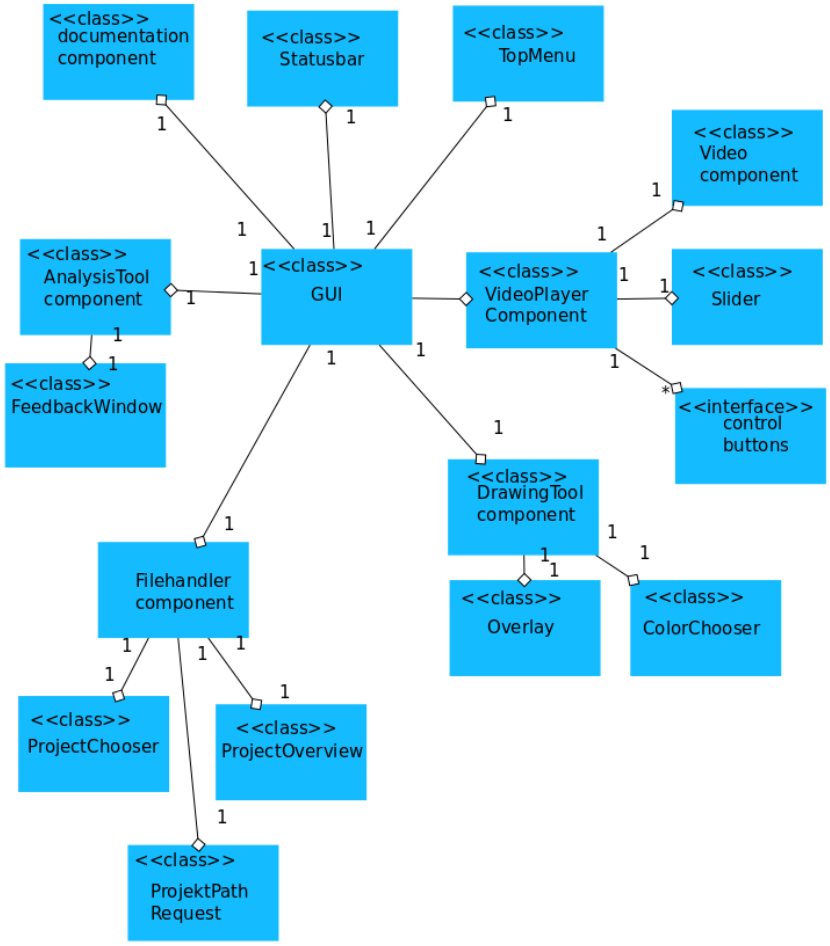
Videospelaren följer ett observatörmönster för att underlätta kontroll av flera filmer samtidigt, se Figur 6.



Figur 6: Diagram som visar observatörmönstret i videospelaren.

5.4 GUI

GUI:t är uppdelat i ett antal komponenter, se figur 7. Samtliga av dessa bör följa designmönstret MVC, se rubrik 3.2.



Figur 7: Diagram för GUI:ts delkomponenter.

6 VERKSAMHETSMÄSSIGT

Detta avsnitt beskriver de verksamhetsmässiga förutsättningarna för programmet så som vilka trådar programmet använder och vilken miljö som används.

6.1 Trådar

Eftersom GUI:t ska gå att kontrollera samtidigt som analys körs måste körningen av GUI:t separeras från analysen. Videospelaren måste också vara tillgänglig under körning av analys så den måste också separeras i körning. Av denna anledning föreslås det att analyser kör på en gemensam tråd och att GUI:t och videospelaren kör på egna trådar.

7 ERBJUDNA GRÄNSSNITT

Detta avsnitt beskriver de gränssnitt som erbjuds av samtliga komponenter.

7.1 GUI

GUI:t ska erbjuda tillgång till att skapa komponenter enligt nedan.

Funktion	Beskrivning	Tillgänglighet
MessageWindow	Visar ett meddelandefönster till användaren.	Samtliga komponenter
ProgressBar	Binder vissa data till ett fönster som uppdateras kontinuerligt.	Analys
PathRequest	Öppnar ett utforskarfönster för att låta användaren välja mapp.	Filhanteringssystem

7.2 Analysmodul

Analysmodulens gränssnitt blir relativt enkelt och konkret. De tillgängliga instruktionerna ska vara generella instruktioner för en godtycklig analys.

Funktion	Beskrivning	Tillgänglighet
startAnalysis	Givet ett analysnamn och en videofil, utför analys på video.	GUI
stopAnalysis	Givet en analys, spara undan analysen och avbryta denna.	GUI
pauseAnalysis	Givet en analys, spara undan analys och pausa denna.	GUI
notify	Informera att analys färdig.	GUI

7.3 Videospelare

Videospelarens erbjudna gränssnitt blir enbart till VideoController som manipulerar de filmer som kör för tillfället, se figur 6.

Funktion	Beskrivning	Tillgänglighet
playPOI	Spela de delar av en film som markerats som POI.	GUI
stepPOI	Hoppa till nästa POI i en film.	GUI

setSpeed	Byt spelhastighet på en film.	GUI
playVid	Börja spela given film (enligt videoinställningar).	GUI
pauseVid	Stanna videon vid nuvarande tidpunkt/bildruta.	GUI
setTimeVid	Hoppa till given tidpunkt i filmen.	GUI
setFrameVid	Hoppa till given bildruta i filmen.	GUI
stepFrame	Hoppa till nästa bildruta i video.	GUI

7.4 Filhanteringssystem

Filhanteringssystemets gränssnitt blir huvudsakligen manipulering av projekt och arbetsmiljö men även sparande av analysinformation och ritningar. Modulen ska också hantera rapportgenerering.

Funktion	**Beskrivning**	**Tillgänglighet**
openProject	Öppnar filmprojekt för display.	Videospelare
getProject	Returnera projektfiler.	Videospelare
saveProject	Spara givet projekt.	GUI, Videospelare
saveAnalysis	Spara analys i givet projekt.	Analys
saveDrawing	Spara ritning i givet projekt.	GUI
exportToDoc	Spara kommentar i dokumentation.	GUI
makeImage	Kombinera ritning och bild till en sammansatt bild.	GUI
writeComment	Kommentera bild.	GUI
generateReport	Generera rapport givet dokumentation.	GUI

7.5 Ritverktyg

Ritverktygets gränssnitt blir huvudsakligen de ritoperationer som ska vara möjliga men också markering av område.

Funktion	**Beskrivning**	**Tillgänglighet**
markArea	Markera område.	GUI
enable/disable Write	Aktivera/avaktivera ritning.	GUI
draw	Rita på given position.	GUI
drawRect	Rita rektangel i givet område.	GUI
drawCircle	Rita cirkel i givet område.	GUI
drawArrow	Rita pil i givet område.	GUI
writeText	Skriv text i givet område.	GUI
setColor	Välj ritfärg.	GUI

8 REFERENSER

[1] Eclipse Foundation, ”Artifact: Architecture Notebook,” [Online]. Hämtat från: http://epf.eclipse.org/wikis/openup/practice.tech.evolutionary_arch.base/workproducts/architecture_notebook_9BB92433.html?nodeId=3178b445. [Använd 30 01 2017].

[2] K. Nilsson, ”Kravspecifikation,” Linköping, 2017.

[3] Wikipedia Foundation Inc, ”Artifact: Observer Pattern,” [Online]. Hämtat från: https://en.wikipedia.org/wiki//Observer_pattern. [Använd 13 02 2017].

[4] Wikipedia Foundation Inc., ”Artifact: Model-view-controller,” [Online]. Hämtat från: https://en.wikipedia.org/wiki/Model-view-controller. [Använd 11 02 2017].

[5] Wikipedia Foundation Inc., ”Artifact: Strategy Pattern,” [Online]. Hämtat från: https://en.wikipedia.org/wiki/Model-view-controller. [Använd 11 02 2017].

[6] J. Sugrue, ”Observer Pattern Tutorial,” 03 02 2010. [Online]. Hämtat från: https://dzone.com/articles/design-patterns-uncovered. [Använd 13 02 2017].

[7] A. Rosenbloom, ”Computer Science University of Toronto, Class Diagrams,” [Online]. Hämtat från: http://www.cs.toronto.edu/~arnold/290/08s/lectures/classDiagrams/. [Använd 11 02 2017].

9 BILAGOR

A. Systemanatomi