

Tredimensionell händelse- och kartvisualisering

Three dimensional event- and map visualisation

**Maya Henriksson Björklund
Filip Josefsson
Daniel Kouznetsov
Tony Lindbom
Simon Magnusson
Viktor Norgren
Gustav Peterberg
Anna Zhao**

Handledare : Henrik Henriksson
Examinator : Kristian Sandahl

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innehåller rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.

Maya Henriksson Björklund

Filip Josefsson

Daniel Kouznetsov

Tony Lindbom

© Simon Magnusson

Viktor Norgren

Gustav Peterberg

Anna Zhao

Sammanfattning

Följande rapport beskriver implementationen av projektet Smart City, ett system som visuellt representerar händelser i realtid i Linköping. Dessa händelser hämtades från öppna datakällor som exempelvis Polisen och SMHI. Händelserna representerades på en tredimensionell karta över staden. De visuella delarna av systemet utvecklades med hjälp Unreal Engine 4 och Cesium. Som databas användes PostgreSQL och en Flask-server användes för kommunikation mellan systemets moduler.

Produkten var tänkt att ge användaren en tydlig överblick av de händelser som skett för att kunna ge en tydligare helhetsbild av staden, detta kunde tänkas användas av exempelvis intresserade privatpersoner, larmoperatör eller stadsplanerare. I slutändan riktade sig produkten mer till privatpersoner. En viktig del av produkten var att enkelt kunna lägga till informationskällor som användaren kan tänkas vilja ha. För att utveckla produkten användes en kombination av Scrum och Kanban. Resultatet blev en produkt som visualiseras ett basutbud av händelser i Unreal Engine 4.

Projektet utfördes under kursen *TDDD96 – kandidatprojekt i programvaruutveckling* på Linköpings universitet. En grupp på åtta medlemmar utvecklade en produkt som grundar sig i en marknadsundersökning. Utöver en rapport som beskriver Smart City hittas även åtta individuella rapporter som undersöker områden kring projektet.

Tillkännagivanden

Författarna av denna rapport vill tacka gruppens handledare Henrik Henriksson som under hela projektet har bidragit med nyttig feedback, idéer och synpunkter. Utan Henrik hade projektet inte blivit detsamma.

Gruppen vill även ägna ett tack åt andra projektgrupper för att de tagit sig tiden att opponera på denna rapport, examinator Kristian Sandahl för väl en genomförd kurs och sektionscaféerna Byttan och Baljan som försett gruppen med hälsosamma mängder koffein under projektets gång.

Innehåll

Sammanfattning	iii
Författarens tack	iv
Innehåll	v
Figurer	viii
Tabeller	ix
1 Introduktion	1
1.1 Motivering	1
1.2 Syfte	2
1.3 Frågeställningar	2
1.4 Avgränsningar	2
1.5 Kontext	2
1.6 Ordlista	2
2 Bakgrund	4
2.1 Idén	4
2.2 Produktens bakgrund och syfte	5
2.3 Marknadsanalys	5
2.4 Gruppens tidigare erfarenheter	5
3 Teori	6
3.1 Programmeringsspråk	6
3.2 Verktyg	7
3.3 Arbetsmetoder	8
4 Metod	11
4.1 Marknadsanalys	11
4.2 Utvecklingsmetoder	12
4.3 Utvecklingsverktyg	13
4.4 Ansvarsområden	14
4.5 Dokument	15
4.6 Metod för att fånga erfarenheter	15
4.7 Kvalitetscoachning	16
5 Resultat	17
5.1 Marknadsanalys	17
5.2 Systemanatomy	18
5.3 Systembeskrivning	19
5.4 Produktutvärdering	25

5.5	Processbeskrivning	25
5.6	Mätning av kvalitetskrav	26
5.7	Gemensamma erfarenheter	26
5.8	Översikt över individuella bidrag	29
6	Diskussion	31
6.1	Resultat	31
6.2	Metod	33
6.3	Arbetet i ett vidare sammanhang	33
7	Slutsatser	35
7.1	Vilka funktioner i Smart City kan skapa värde för marknaden?	35
7.2	Vilka erfarenheter kan dokumenteras från detta projekt och kan vara intressanta för framtida projekt?	35
7.3	Vilka utvecklingsmöjligheter finns till Smart City som skulle öka marknadsvärdet?	36
7.4	Vad för värde skapar en systemanatomi för ett projekt i denna storlek?	36
A	Att agera på aggregerad data - Maya Henriksson Björklund	37
A.1	Introduktion	37
A.2	Teori	37
A.3	Bakgrund	38
A.4	Metod	38
A.5	Resultat	39
A.6	Diskussion	42
A.7	Slutsatser	43
B	Unreal Engine: Mer än bara underhållning - Filip Josefsson	44
B.1	Introduktion	44
B.2	Teori	45
B.3	Metod	45
B.4	Resultat	46
B.5	Diskussion	49
B.6	Slutsatser	51
C	Spelmotorers påverkan på designprocessen - Daniel Kouznetsov	53
C.1	Introduktion	53
C.2	Teori	53
C.3	Metod	54
C.4	Resultat	55
C.5	Diskussion	58
C.6	Slutsatser	58
D	Metodik vid framtagning av krav - Tony Lindbom	60
D.1	Introduktion	60
D.2	Teori	60
D.3	Metod	65
D.4	Resultat	66
D.5	Diskussion	69
D.6	Slutsatser	70
E	Hur uppmärksammar man en användare när det finns oändligt med information - Simon Magnusson	72
E.1	Introduktion	72

E.2	Teori	73
E.3	Metod	73
E.4	Resultat	74
E.5	Diskussion	76
E.6	Slutsatser	77
F	Hur kan Q-learning användas för att testa ett användargränsnärt utvecklat i Unreal Engine 4? – Viktor Norgren	79
F.1	Introduktion	79
F.2	Teori	79
F.3	Metod	81
F.4	Resultat	82
F.5	Diskussion	85
F.6	Slutsatser	85
G	Analysmetoder för att lokalisera flaskhalsar i mindre mjukvarusystem likt Smart City - Gustav Peterberg	87
G.1	Introduktion	87
G.2	Teori	88
G.3	Metod	89
G.4	Resultat	89
G.5	Diskussion	95
G.6	Slutsatser	97
H	Versionshantering av Unreal Engine 4 projekt med Git- Anna Zhao	98
H.1	Introduktion	98
H.2	Syfte	98
H.3	Frågeställning	98
H.4	Teori	98
H.5	Metod	100
H.6	Resultat	101
H.7	Diskussion	103
H.8	Slutsatser	104
I	Övriga tabeller	105
	Litteratur	107

Figurer

5.1	<i>Systemöversikt</i>	20
5.2	<i>Bild från en workshop</i>	21
5.3	<i>Exempel på svar från server</i>	22
5.4	<i>Databasstruktur</i>	23
5.5	<i>Systemets visualiseringsträäd</i>	24
5.6	<i>Systemets arkitektur i Unreal Engine 4</i>	24
5.7	<i>Bild från den resulterade produkten</i>	25
C.1	<i>Diagram på processen för visualisering i UE.</i>	56
C.2	<i>Visualisering av bostad.</i>	56
F.1	<i>Användargränssnittet för den egna implementationen</i>	84
G.1	<i>Exempel på en operationsväg genom ett system.</i>	92
G.2	<i>Översikt av operationsträd</i>	94
H.1	<i>Binärfil öppnad i en textredigerare.</i>	99
H.2	<i>Ett exempel av en arbetsgång där merge används.</i>	100

Tabeller

5.1	<i>Medelbetyg för varje fråga för varje sprint.</i>	26
D.1	<i>Tabell som visar för- och nackdelar för olika konversationsmetoder</i>	67
D.2	<i>Tabell som visar för- och nackdelar för olika observationsmetoder</i>	67
D.3	<i>Tabell som visar för- och nackdelar för olika analytiska metoder</i>	68
D.4	<i>Tabell som visar för- och nackdelar för olika syntetiska metoder</i>	68
D.5	<i>Tabell som visar för- och nackdelar hos marknadsanalys</i>	69
D.6	<i>Tabell som visar för- och nackdelar hos frågeformulär</i>	69
G.1	<i>Påhittade potentiella resultat efter mätningar av exekveringstider på operationsvägar genom figur G.1</i>	93
G.2	<i>Uppdelning och namngivna funktioner med dess tillhörande värde från resultatet i tabell G.4.2</i>	93
I.1	<i>Projektets kravstatus.</i>	105



1 Introduktion

Idag lever människor i en hektisk värld där mycket händer varje dag. Många plattformar samlar data men dessa är nödvändigtvis inte samlade på ett ställe. Om en person vill få reda på information måste även dessa plattformar användas. Detta projekt, *Smart City*, ämnar sig till att samla data på samma ställe. Informationen samlas från olika händelser i staden som till exempel: antal cyklar som åker förbi på en plats, vägarbeten och väder.

I dagsläget finns det få produkter som samlar flera informationskällor och visualiseringar dem. Dessa produkter använder sig av tvådimensionella kartor och väldigt unika informationskällor för ett visst användningsområde. Projektgruppen såg där möjligheten att förbättra visualiseringen av händelser genom att presentera dem i en tredimensionell karta. Men även att göra det enkelt att lägga till fler informationskällor. Projektets avsikt var att leverera Smart City som öppen källkod.

I detta kapitel redovisas projektets syfte och motivering samt rapportens frågeställningar.

1.1 Motivering

Projektgruppens vision med produkten var att visuellt presentera aggregerad data för att skapa en samlad överblick av händelser. Detta skulle göra det möjligt att behandla större mängder information och skapa en helhetsuppfattning för användaren. Genom att utveckla produkten på ett sätt som gjorde det lättare att lägga till informationskällor kan den data som presenteras i produkten bli skräddarsydd för användaren.

En viktig del av produkten var att den presenterade händelser i staden i tre dimensioner. Detta gjorde att produkten kunde sticka ut på den befintliga marknaden. Genom att konstruera produkten för att vara modulär kunde systemet enkelt utökas med fler datakällor och på så vis presentera mer information för användaren.

Genom att läsa denna rapport kan läsaren skapa sig en uppfattning hur information i en stad kan samlas och visuellt representeras digitalt och vilka framtidsmöjligheter det finns för denna produkt. Läsare kan även ta del av den erfarenhet projektgruppen erhöll under projektets gång som kan vara användbar för vidare utveckling av produkten.

1.2 Syfte

Syftet med projektet var att ge projektgruppen erfarenhet och lärdomar av att utveckla en mjukvaruproduct baserat på krav från en marknadsanalys.

Rapportens syfte är att beskriva det resulterande systemet och samla de erfarenheter, beslut och lärdomar som fångats upp genom arbetet med projektet. Projektmedlemmarnas individuella rapporter tar även upp områden eller verktyg relaterade till projektet. Rapporten kan användas som grund för vidare utveckling av produkten Smart City.

1.3 Frågeställningar

1. Vilka funktioner kan göra Smart City attraktivt för marknaden?
2. Vilka erfarenheter kan dokumenteras från detta projekt och kan vara intressanta för framtida projekt?
3. Vilka utvecklingsmöjligheter kan implementeras i Smart City som skulle öka marknadsvärdet?
4. Vad för värde skapar en systemanatomy för ett projekt av denna storlek?

1.4 Avgränsningar

Smart City konstruerades för att endast ta emot information. Det innebär att Smart City inte kan användas för att kommunicera utåt för att till exempel ge respons på händelser. Systemet är menat för att endast visualisera data från olika datakällor.

Produkten har möjlighet att presentera händelser över hela jordgloben. För detta projekt valde projektgruppen att visualisera informationskällor från endast staden Linköping, där produkten utvecklades.

Rapporten beskriver inte genomgående, tekniska detaljer om implementationen utan ger endast en övergripande bild av systemet.

1.5 Kontext

Projektet ingick i kursen *TDDD96 – Kandidatprojekt i programvaruutveckling* vid Linköping universitet. Projektgruppen bestod av åtta medlemmar som läste programmen *civilingenjör i mjukvaruteknik* och *civilingenjör i datateknik*.

Att projektet utfördes i en kurs på universitetet medförde olika deadlines och inlämningar som skulle uppfyllas och godkännas. Varje projektmedlem hade enligt kurskrav 400 timmar att disponera på utveckling av produkten, rapportskrivning, och individuell rapportskrivning.

Gruppen valde att i slutet av projektet lägga ut produkten som öppen källkod enligt Apache 2.0 för allmän användning.

1.6 Ordlista

Discord

Webbaserad plattform för kommunikation via text och röst. På plattformen finns det möjlighet att skapa flera kanaler i en server för skilda text- och röstchattar. [1]

Messenger

Webb- och mobilbaserad chatt skapad av Meta Platforms, Inc [2].

InfluxDB

En tidsseriedatabas med öppen källkod [3].

REST

Representational State Transfer, är en standard för hur internettförfrågningar ska hanteras [4].

REST API

Ett API som förhåller sig till REST-standarden [5].

Tidsseriedatabas

En databas där värdena i databasen ordnas efter tid [6].



2 Bakgrund

I detta avsnitt beskrivs bakgrunden för projektets början och de första utgångspunkterna för dels projektmedlemmarna och produkten.

2.1 Idén

Grundidén till projektet var att skapa ett system som samlar in information från en stad och visualiseras detta på en virtuell karta. Produkten skulle rikta sig mot att agera kontrolltorn över till exempel larmuttryckningar. Denna information skulle presenteras i form av händelser som dyker upp i en lista som ger ytterligare information om händelserna. Händelserna i listan skulle även korrelera med markörer på kartan som visar var i staden det skett. Exempel på vad händelser skulle kunna vara var vägarbeten och väder.

Nya händelser som skedde skulle dyka upp på kartan i realtid för att överblicken av staden ska vara så nära verkligheten som möjligt. Användaren skulle kunna filtrera informationen som presenterades för att fokusera på enskilda typer av händelser. En tidslinje skulle också finnas för att kunna gå tillbaka och ge en återblick över vad som skett. Planen var även att i mån av tid möjlig implementera en AI som skulle agera övervakare över dessa händelser.

För att utveckla systemet var tanken att använda Unreal Engine 4 (*UE4*) för visualisering av händelser. Dessa händelser skulle lagras i två databaser, PostgreSQL och InfluxDB. Dessa idéer grundar sig i ett projekt vid Linköpings universitet som hölls av biträdande professor Magnus Bång. Initialt var tanken att detta skulle vara ett delsystem till ett externt projekt i större storlek.

Projektgruppen valde att behålla grundidén som utgångspunkt för projektet. Dock ändrades produktens syfte till att rikta sig mer till allmänheten och privatpersoner för visualisering av godtyckliga händelser i till exempel en stad. Från grundidén valde Projektgruppen att implementera produkten med en databas, PostgreSQL och att i mån av tid lägga till InfluxDB. Projektgruppen valde att behålla idén att utveckla produkten i UE4.

2.2 Produktens bakgrund och syfte

Produkten grundade sig i en marknadsanalys utförd av projektgruppen. Från analysens resultat skrevs en kravspecifikation med syftet att skapa en produkt med marknadsvärde. Produkten utvecklades för att levereras som öppen källkod. Den initiaла planen var att färdigställa projektet med professorn som beställare. Däremot beslutade projektgruppen av särskilda skäl att avsluta samarbetet innan utveckling av produkten hade börjat.

2.3 Marknadsanalys

För att ta fram de funktionella krav som sattes på produkten gjordes en marknadsanalys. För mer information se kapitel 4.1.

2.4 Gruppens tidigare erfarenheter

Samtliga projektmedlemmar hade akademiska erfarenheter av grupperbete och mjukvaruutveckling. Vissa hade även arbetslivserfarenhet av mjukvaruutveckling sedan innan projektet.

Studenterna som läste data teknik hade erfarenheter från bland annat projekt kurserna *TSEA29 – Konstruktion med mikrodatorer* medan studenterna som läste mjukvaruteknik hade liknande erfarenheter från kursen *TDDD92 – Artificiell intelligens - projekt*. Dessa kurser har givit gruppmedlemmarna kunskaper inom programutveckling, programmering och större projektarbeten. Samtliga studenter har läst kursen *TDDC93 – Programutvecklingsmetodik, teori* som gett medlemmarna teoretisk kunskap om bland annat agila arbetsmetoder.

Under tidigare projekt fick medlemmarna erfarenheter som att ha en gemensam projektkalender och att bibehålla god kommunikation med varandra genom hela projektet. I de projekten hade vissa medlemmar upplevt att dålig kommunikation lett till att gruppmedlemmar inte vet hur projektet ligger till och att det gemensamma målet glömts bort.

Ingen av gruppmedlemmarna hade tidigare erfarenhet av att jobba med Unreal Engine 4.



3 Teori

Detta avsnitt beskriver den underliggande teorin som används vid utvecklingen, bland annat utvecklingsmiljöer och verktyg.

3.1 Programmeringsspråk

Detta kapitel tar upp de programmeringsspråk gruppen använt sig av.

3.1.1 Python

Python är ett strikt dynamiskt typat objektorienterat högnivåspråk. Språket är interpretat vilket tar bort kravet för kompilering vid exekvering. För att bibehålla att kod skall vara lättläslig i Python har dokumentation skrivits av skaparna av Python för att visa hur Pythonkod ska skrivas. PEP 8 är den praxis över hur detta ska genomföras. [7]

3.1.2 C++

C++ är ett kompilerat och starkt typat programmeringsspråk. C++ stödjer också objektorientering. Programmeringsspråket ger möjligheten att programmera nära hårdvaran men också implementera programmeringsparadigmer på högre nivåer. Högre nivåer betyder mer abstraktion, *templates* används för att uppnå detta genom att återanvända funktioner. [8, 9]

3.1.3 SQL

SQL är ett språk för manipulering och hantering av databaser. SQL används för att efterfråga, uppdatera och skapa ny data i en databas. [10]

PostgreSQL

PostgreSQL är en relationsdatabas som grundar sig i SQL-standarden. En av de funktioner som det tillhandahåller är möjligheten att skapa egna datatyper. [11]

3.2 Verktyg

Denna del av rapporten tar upp de verktyg som gruppmedlemmarna har använt vid utveckling av produkten.

3.2.1 Unreal Engine 4

UE4 är en spelmotor med många verktyg som ger möjligheten att skapa applikationer, även applikationer som inte klassas som spel. Motorn erbjuder en stabil grund för grafik som underlättar utveckling av tredimensionella miljöer. De primära programmeringsspråken som motorn stödjer är C++ och *Blueprints Visual Scripting*. UE4 har en uppsättning inbyggda bibliotek som stödjer en rad funktioner, till exempel finns stöd för HTTP-anrop och JSON-hantering. UE4 har även möjligheten till att utöka funktionalitet genom tredjepartstillägg. Spelmotorn är nära integrerad med utvecklingsmiljön Visual Studio för utveckling av kod. [12]

Blueprints Visual Scripting

Visuellt skriptspråk som används i UE4. Skriptspråket är nodbaserat där olika noder kopplas ihop med olika funktioner för att skapa ett önskat beteende. Språket kan ses som ett komplement till mer traditionella skriptspråk. Förkortat kallas språket för *Blueprints*. [13]

Cesium

Cesium är ett verktyg med öppen källkod som gör det möjligt att implementera 3D-kartor i UE4. Verktyget ger höjdkartor, satellitbilder och 3D-representationer av byggnader. [14]

3.2.2 Visual Studio

Visual Studio är en utvecklingsmiljö från Microsoft för mjukvaruutveckling i flertalet programmeringsspråk. Visual Studio har tilläggspaket som ger stöd för de flesta programmeringsspråk och bibliotek. Till exempel finns stöd för C++ och bibliotek för spelutveckling i C++. [15]

3.2.3 Git

Git är ett verktyg som låter användare spara och versionshantera filer samt se historik. Verktyget möjliggör att flera användare samtidigt kan arbeta med koden. Kod laddas upp (*pushes*) till ett *repository* där användare kan hämta koden för att modifiera och ladda upp ändringar. Kod som läggs upp på Git läggs upp genom att göra en *commit* till projektets *Repository*. För att dela upp arbetet under utveckling kan en gren (*branch*) skapas för göra isolerade ändringar innan det laddas upp till huvud-*branchen*. [16]

GitLab

GitLab är ett webbaserat verktyg som utökar Git med bland annat ett grafiskt gränssnitt och *pipelines* som tillåter automatiserad testning. [17]

Git LFS

Git Large File Storage ersätter stora filer med textfiler som har en pointer i sig och lagrar den stora filens innehåll på en extern server som till exempel GitHub.com. [18]

Git feature branches

Git feature branches är en arbetsgång i GitLab där olika funktioner av projektet delas upp och utvecklas i olika branches. När utvecklingen är klar i en feature-branch *merge:as* den in till huvud-branchen. [19]

3.2.4 Black

Black är ett verktyg som bland annat formatterar pythonkod enligt PEP 8-standarden. Verktyget har även andra funktioner som kontroll av andel kommenterade funktioner. [20]

3.2.5 Pytest

Pytest är ett testramverk skapat för att utföra tester, testerna implementeras i Python. Dessa tester skapas av utvecklare och anpassas efter vad funktioner ska utföra. Med hjälp av testfilter kan Pytest kontrollera om en programkoden returnerar giltiga värden. När testerna är skapade kan de appliceras på Python-skript och skriva ut en sammanställning av testerna i till exempel kommandotolkten. Testerna kan integreras med Gitlabs pipelines som då utför tester på all kod som läggs upp. [21]

3.2.6 Flask

Flask är ett ramverk för Python som används för att implementera webbapplikationer. Flask är designat att ha få beroenden till andra bibliotek. Flask använder sig av *Web Server Gateway Interface* vilket är ett standardgränssnitt mellan webbservrar och webbapplikationer också skrivet i Python. [22]

3.2.7 Ramverk

Med ramverk menas en samling av moduler och bibliotek som tillåter enklare utveckling på hög abstraktionsnivå.

3.3 Arbetsmetoder

Detta kapitel beskriver arbetsmetoder som användes i projektet.

3.3.1 Scrum

Scrum är ett agilt och iterativt arbetsätt som används inom till exempel mjukvaruutveckling men är generell för produktutveckling. Scrum baseras på att projektet delas in i mindre delar som utförs under så kallade *sprints*. Dessa sprints pågår under en viss tidsperiod och utförs enligt en förbestämd process. Inför varje sprint kan ett möte hållas där målen med den kommande sprinten tas upp, vad som ska göras och vem som ska göra vad. Efter varje sprint utförs en *sprint-återblick* vilket häданefter kallas *utvärderingsmöte*. På mötet lyfts bland annat vilka områden som fungerat bra för gruppen och vilka som behövde förbättras. [23, 24]

Inom Scrum finns en *Scrum-mästare* som leder de moment som tillhör Scrum [24]. Scrum-mästaren har som ansvar att hålla koll på att sprinten innehåller de moment som behöver utföras, ansvarar för att utvärdering hålls samt ser till att de processförbättringar som behövs göras faktiskt genomförs.

3.3.2 Kanban

Kanban är ett ramverk för att implementera en agil arbetsgång för bland annat mjukvaruutveckling. Ramverket går ut på att ha en transparent arbetsgång som representeras visuellt på Kanban-tavlor. Alla medlemmar kan på tavorna se status på olika arbetsområden och vem som jobbar med vad. På dessa tavlor satte gruppmedlemmarna upp vad som behövde göras, vad som jobbades på för tillfället och vad som var färdigställt. [25]

3.3.3 Trello

Trello är ett webbaserat verktyg för att hantera och strukturera upp arbetsuppgifter. Användare kan logga in och skapa tavlor där andra användare kan ansluta sig. En tavla innehåller listor med kort skapade av de som är med på tavlan. Korten visualiseringar arbetsuppgifter och kan delas upp i olika kategorier efter behov. [26]

3.3.4 Marknadsanalys

En marknadsanalys är en analys utförd inom en marknad för att hämta information om vilka produkter som finns och vilka leverantörer som existerar på den. Detta kan användas vid köp av produkter eller vid utveckling av produkter som ska ut på marknaden. Analysen kan utföras stegvis enligt utdraget från upphandlingsmyndigheten nedan. [27]

Kartlägga marknaden

Här nedan är ett utdrag från upphandlingsmyndigheten över potentiella frågor som kan vilja besvaras.

- Vilka är leverantörerna på marknaden?
- Vad erbjuder leverantörerna utifrån det behov ni har?
- Kan leverantörerna på marknaden uppfylla kraven?
- Hur ser utvecklingen på marknaden ut?
- Finns det några branschspecifika utmaningar och hur kan dessa i så fall hanteras?

Olika sätt att samla in information

Analysen kan utföras via den information som finns tillgänglig på internet eller allmänt känd information. Ibland krävs dock en djupare analys för att skaffa tillräcklig kunskap. Nedan kommer en lista över andra potentiella sätt att hämta information.

- Kontakta olika branschorganisationer och experter inom respektive område.
- Rådfråga andra kommuner, regioner eller myndigheter som har liknande behov.
- Gå igenom egna och andra organisationers upphandlingsdokument.
- Gå igenom eventuella rättsfall inom området.
- Besöka mässor, delta i olika nätverk.

Modeller och verktyg

En marknadsanalys kan även utföras med olika modeller och verktyg. Till exempel finns Porters femkraftsmodell och PESTEL.

Porters femkraftsmodell

Utdrag ur upphandlingsmyndighetens sida, "Porters femkraftsmodell är ett vanligt verktyg inom olika former av marknadsinriktad omvärldsanalys och som bland annat kan användas för att beskriva hur attraktiv en marknad är för att verka på för ett visst företag.".

PESTEL

Utdrag ur upphandlingsmyndighetens sida, "Pestel är en strategisk analysmodell för att identifiera makrofaktorer som påverkar ett företag. De fyra makrovariablerna är: politiska, ekonomiska, sociala och teknologiska. Analysen är användbar för att studera de externa faktorerna i en SWOT-analys."

Dialog med marknaden

En grundligt utförd marknadsanalys innehåller någon form av dialog med potentiella leverantörer. Detta kan ske genom enskilda dialogmöten med leverantörer eller genom begäran av information.

Analysera resultaten

Efter informationssamlingen ska marknaden fått en bättre helhetsbild. Nu kan resultaten analyseras och besvara frågor se exempel från upphandlingsmyndighetens sida nedan.

- Lösningar på marknaden som tillgodosser behovet
- Behov av nya lösningar eller förbättringar
- Behov av att leverantörer utvecklar eller forskar fram innovativa lösningar.



4 Metod

I detta kapitel delges de metoder som användes för att genomföra projektet och samla erfarenhet.

4.1 Marknadsanalys

För att ta fram de krav som används i *kravspecifikationen* för produkten *Smart City*, utfördes en marknadsanalys. Marknadsanalysen utfördes i fem steg enligt riktlinjer från upphandlingsmyndigheten som beskriver hur en generell marknadsanalys utförs. Nedan kommer en lista som presenterar de fem stegen som användes i projektet.

1. Marknadsanalysen började med en produktidé, idén är en grund för att kunna börja göra en undersökning i steg 2.
2. Marknaden undersöktes för att hitta konkurrens och liknande produkter.
3. Utifrån steg 3 undersökts de olika produkterna genom att läsa vad de hade att erbjuda till sina kunder, dessa egenskaper listas och analyseras.
4. Sedan jämfördes de olika egenskaperna för att se vad som skulle kunna utvecklas i grundidén eller förbättras för att sticka ut mer på den fria marknaden.
5. Utifrån analysen i steg 4 togs egenskaper fram som skulle särskilja produkten på marknaden.

Steg 1: I Första steget hade projektgruppen en workshop för att diskutera fram en grund till produkten utifrån en extern produktidé. Se kapitel 2.1 för hur projektgruppen valde produktidé.

Steg 2: Efter att idén utformades utfördes en undersökning på internet efter liknande produkter. Detta gjordes genom att söka på Googles sökmotor med söktermer som till exempel de listade nedan.

- User maps
- Event maps
- Statistical maps

- 3D maps

Från sökresultaten undersöktes de hemsidor som listades varpå de produkter som var liknande Smart City valdes ut. Produkten valdes om den ansågs ha syfte och egenskaper likt projektidén. De produkter som redan fanns på marknaden sammanfattades i ett dokument för att sedan lättare kunna gå djupare i undersökningen om produkterna. I detta dokument skrevs en liten sammanfattningsom produkten och företaget som levererar den.

Steg 3: När konkurrensen undersökts togs för- och nackdelar hos varje produkt fram efter projektgruppens egna värderingar gentemot projektets syfte. Därefter sammansättades de viktigaste punkterna

Steg 4: Utifrån steg 3 valdes olika egenskaper ut som systemet skulle fokusera på. För att välja ut dessa egenskaper hade projektgruppen en workshop där alla egenskaper diskuterades och värdesattes enligt projektmedlemmarnas åsikter. Dessa egenskaper skulle skilja Smart City från nuvarande produkter för att sticka ut på marknaden.

Steg 5: Utifrån egenskaperna som togs fram i steg 4 diskuterade projektgruppen de krav som tycktes vara viktigast att framhäva för att sedan kunna bygga vidare på. Dessa krav låg sedan till grund för de krav som användes till *Smart City*.

4.2 Utvecklingsmetoder

Projektgruppen valde att arbeta enligt en kombination av Scrum och kanban för att utveckla dokument och produkten. Webbaserade Trello användes som kanbantavla för att strukturera upp aktiviteter för sprintar. För att versionshantera produkten och dokument användes Gitlab där arbetet delades upp i flera projekt.

4.2.1 Scrum

Under projektet arbetade gruppen enligt den agila arbetsmetoden Scrum. I detta projekt hade sprinten en längd på en vecka. Innan varje sprint diskuterades vad som behöver göras kommande sprint. I projektet valdes aktiviteter av projektmedlemmarna själva, detta för att arbetet blev mer flexibelt men samtidigt fanns en överblick av vad som arbetades på. Under sprinten förekom möten där gruppmedlemmarna kort gav en uppdatering om hur arbetet gick, dessa möten hölls dock inte varje dag utan de dagar som medlemmarna samlades på plats för att arbeta, vilket var i genomsnitt tre dagar i veckan.

Sprinten avslutades med ett utvärderingsmöte där den gångna sprinten utvärderades muntligt och med hjälp av enkäter. Innan mötet skulle gruppmedlemmarna fylla i en Google Forms enkät för att samla erfarenheter och idéer för processförbättring. Mötet hölls av Scrum-mästaren som tog upp resultatet från senaste enkäterna och lyfte de idéer som projektgruppen antecknat för diskussion. Resultatet från utvärderingsmötet användes sedan i kommande sprint.

Första delen av frågorna i enkäten skulle skattas från 1 till 7:

- Hur väl nåddes de mål som sats upp under denna sprint?
- Hur känner du att din insats i projektet varit denna sprint?
- Hur upplevdes senaste sprinten?
- Kände du att tiden räckte till?

Andra delen av enkäten var frågor med frisvarstext:

- Vad gick bra denna sprint?
- Vad gick inte så bra denna sprint?
- Vad har vi lärt oss?
- Vad har vi kvar att lösa?
- Processförbättringsförslag

Genom att gå igenom svaren på utvärderingsmötena kunde gruppen tillsammans diskutera vad som borde ändras i arbetsprocessen för att effektivisera arbetet.

4.2.2 Kanban

Gruppen använde tjänsten Trello för att skapa och använda en Kanban-tavla. Kanban-tavlorna användes för att strukturera upp vad som skulle göras i den aktuella sprinten och gjorde det tydligt för alla medlemmar att se projektets status. Scrum-mästaren satte upp Kanban-tavlor för varje sprint och projektmedlemmarna valde själva ut de aktiviteter som skulle utföras.

4.2.3 Parprogrammering

Vid utveckling av områden i UE4 användes parprogrammering, detta utfördes antingen på plats eller på distans. Vid distansarbete med parprogrammering användes Discord som plattform. Två ur projektgruppen satt då på samma Discord-kanal där en person skärmdelade medan den andre observerade och kom med förslag på lösningar.

4.2.4 Integrationsmöte

Under utvecklingen behövde olika delar av systemet mergeas i Gitlab. Projektgruppen hade vid dessa tillfällen ett integrationsmöte för att slå samman de olika delarna till projektets huvud-branch.

4.3 Utvecklingsverktyg

I detta kapitel beskrivs de utvecklingsverktyg som användes under projektets gång.

4.3.1 Unreal Engine 4

För att utveckla den visuella delen av produkten användes UE4. Projektgruppen använde sig av motorns stöd för C++ och Blueprints för att utveckla den funktionalitet som visas i slutprodukten. Majoriteten av utvecklingen i UE4 gjordes med parprogrammering. Projektgruppen använde sig av *feature branches*.

4.3.2 Versionshantering

För versionshantering användes Git under projektet. Utvecklingen delades upp på GitLab i olika delar för respektive delsystem för att strukturera arbetet. Projektgruppen använde Git LFS för att hantera Blueprints i UE4.

4.3.3 Visual Studio

Visual Studio användes för utveckling i UE4 och för att kompilera C++-kod. Till Visual Studio användes de nedladdningsbara paketen "Desktop development for C++" och "Game development with C++".

4.4 Ansvarsområden

Varje gruppmedlem hade ansvar att utföra de arbetsuppgifter som förväntades inom en given tidsram. Kunde uppgiften inte utföras inom den givna tidsramen skulle detta meddelas till de andra projektmedlemmarna. Detta för att ge de andra medlemmarna chansen att hjälpa till för att i så stor utsträckning som möjligt undvika förseningar. I projektets början skapades ett gruppkontrakt för att definiera allmänna förväntningar på gruppmedlemmar. Samtliga medlemmar godkände och blev bundna att agera enligt kontraktets riktlinjer projektet ut. Under projektet hade även varje medlem en unik roll, dessa roller var förbestämda av kursen.

Teamledare - Maya Henriksson Björklund

Teamledaren hade som ansvar att driva projektet framåt och att coacha samtliga gruppmedlemmar genom projektet. I detta ingick att hålla övergripande koll på projektet och gruppmedlemmar för att se till att projektet gick som planerat. Teamledaren hade även ansvar att löpande boka in handledarmöten och att bibehålla en god arbetsmiljö.

Analysansvarig – Tony Lindbom

Analysansvarig hade som ansvar att göra en marknadsundersökning för att hitta liknande produkter och samla information om vad som kan göra Smart City attraktiv på marknaden. Analysansvarig hade också ansvaret att sammanställa produktens krav baserat på utförd marknadsanalys samt vid behov uppdatera produktens kravprofil.

Arkitekt – Simon Magnusson

Det var arkitektens ansvar att ta fram en struktur för de komponenter och gränssnitt som skulle finnas i projektet. Det ingick även i arkitektens ansvar att ha övergripande koll på hur dessa komponenter och gränssnitt fungerade och eventuellt hängde ihop. Arkitekten hade även ansvaret över de dokument som rör systemarkitekturen. Arkitekten jobbade tätt med utvecklingsledaren.

Utvecklingsledare – Daniel Kouznetsov

Utvecklingsledaren hade som ansvar att se till att de komponenter som utvecklades var implementerbara i det existerande projektet. Utvecklingsledaren hade även sista ordet när det kom till tidsuppskattnings av utvecklingen och hade rollen som Scrum-mästare. Utvecklingsledaren jobbade tätt med arkitekten.

Testledare – Viktor Norgren

Det var testledaren som hade ansvar för att avgöra vilka delar av projektet som behövde testas och hur dessa skulle testas. Det var även testledaren som skapade alla tester och ansvarade för att informera resten av gruppen hur dessa tester skulle användas.

Kvalitetssamordnare – Filip Josefsson

Kvalitetssamordnaren hade som ansvar att ta fram mätbara kvalitéer som är relevanta för projektet och att sammanställa dessa i en kvalitetsplan. Kvalitetssamordnaren jobbade tätt med testledaren för att utveckla och utföra de tester som krävdes för att säkerställa att satta mål uppnåddes.

Konfigurationsansvarig – Anna Zhao

Konfigurationsansvarig hade ansvaret för att hålla reda på vad som ska versionshanteras och var det sker. Denna roll hade även ansvaret att sätta standarden för hur versionshantering ska se ut, som till exempel hur commit-meddelanden ska skrivas i Git. Målet med detta var att få en sammanställning på gruppens arbetsgång samt spara arbete på en extern plattform som en backup.

Dokumentansvarig – Gustav Peterberg

Dokumentansvarig hade som ansvar att sätta upp standarder för samtliga dokument

och hade som ansvar att se till att alla medlemmar var medvetna om satt standard. Standarder sattes upp genom att använda beprövade mallar. Övriga textstandarder sattes upp likt liknande rapporter med bland annat en lathund för rapportskrivning. Dokumentansvarig hade ansvaret att se till att samtliga dokument följer sagd standard till samtliga inlämningar. Dokumentansvarig skrev inte alla rapporter själv men hade ansvaret över och var med på alla producerade dokument. Dokumentansvarig hade sista ordet i dokumentrelaterade beslut.

4.5 Dokument

Detta kapitel beskriver de dokument som skapats och använts under utveckling av projektet.

- **Projektplan:** Projektplanen hade som syfte att planera och strukturera upp projektet. I denna ingick en plan för utvecklingsfasen av projektet samt en uppskattning av antal timmar per projektmodul. Det ingick även en riskanalys och utbildningsplan. Projektplanen sammanfattade även syfte, avgränsningar och bakgrund till projektet samt en fasplan som beskrev ansvarsområden för samtliga roller.
- **Kravspecifikation:** Kravspecifikationen hade som syfte att sammanställa de krav som tagits fram baserat på en marknadsundersökning. Dessa krav är prioriterade och kategoriseringar gjordes efter användningsområde.
- **Arkitekturplan:** Arkitekturplanens syfte var att lägga en grund för hur utvecklingsfasen av projektet skulle fungera. Arkitekturplanen tar upp hur systemet som byggs upp under projektets gång ska se ut och fungera. Dokumentet tar upp hur moduler ska samverka och hur Unreal Engine 4 systemet ska fungera internt. Utöver detta tas även övergripande designbeslut upp.
- **Testplan:** Testplanen hade som syfte att beskriva för gruppen hur systemet ska testas under projektets gång.
- **Kvalitetsplan:** Kvalitetsplanen tar upp de kvalitetsmål som finns gällande systemet. Den tar även upp hur gruppen ska jobba mot dessa mål samt hur gruppen kontrollerar att dessa mål har uppfyllts.

4.6 Metod för att fånga erfarenheter

Detta kapitel beskriver de huvudsakliga metoder som använts för att fånga gruppmedlemmarnas erfarenheter under projektets gång. Detta gjordes för att dels effektivisera arbetet och dels kunna besvara på rapportens andra frågeställning gällande dokumentervärda erfarenheter. Vad gick bra och vad gick mindre bra genom att använda vissa verktyg eller genom avsaknad av verktyg.

4.6.1 Handledarmötén

Varje vecka hölls ett möte med handledaren vars syfte var att sammanfatta den gångna veckan och ge en status på projektet. På mötena togs även planen för den kommande veckan upp och eventuella synpunkter skrevs ner för att diskuteras internt.

4.6.2 Sprint-enkäter

Innan sprint återblick-mötena fyllde projektmedlemmarna i en enkät i syfte att dokumentera och utvärdera senaste sprinten. Medlemmarna fick svara på frågor som undersöker positiva och negativa delar av sprinten samt produktiviteten hos sprinten. Första delen av enkätens

frågor besvarades med en siffra ett till sju där ett var ”inte alls” och sju motsvarade ”mycket väl”. Frågorna besvaras med siffror för att enklare kunna mäta och i slutändan dra någon slutsats om hur utvecklingen går och var den är på väg. I andra delen av enkäten besvarades frågorna i ord för att ge rum att förklara vad som var positivt eller negativt. Resultaten från enkäten sparades sedan digitalt för att göra det lättare för gruppen att se hur gruppmedlemmarnas tankar under projektet utvecklades. Detta gjordes för att sedan ta fram förbättringsförslag för nästkommande sprint. Resultat på sprintenkäter som genomfördes ses i kapitel 5.5.

4.6.3 Utvecklingsmöten

För att få ut erfarenheter och insikter från sprint-enkäterna hölls ett utvecklingsmöte i slutet av sprinter. Scrum-master skulle hålla mötet och ta upp enkätens svar för att utreda processförbättring eller allmänna tankar kring utvecklingen.

4.6.4 Erfarenhetsmötet

För att samla de erfarenheter som erhålls under projektet valde projektgruppen att i slutet av projektet att sätta sig ned och ha ett erfarenhetsmöte. På mötet skulle alla projektmedlemmar berätta vilka användbara erfarenheter de fått och de tankar som fanns kring projektets utförande.

4.7 Kvalitetscoachning

Under projektet hade gruppen totalt fyra workshoppar som leddes av en studentgrupp på fyra studenter från kursen *TDDE46 - Programvarukvalitet*, på dessa workshoppar närvarade representanter från två andra projektgrupper. Under dessa workshoppar fick de närvarande grupperna feedback på bland annat projektets kvalitetskrav och kvalitetsplanen. Workshopparna var även ett tillfälle att få utbyta information med andra projektgrupper och höra andra studenters erfarenheter inom området. Innan samarbetets slut bedömde de även hur de tyckte projektarbetet hade gått från deras perspektiv i en skriftlig rapport.



5 Resultat

Detta kapitel redovisar de resultat som projektet producerat. För att redovisa resultaten presenteras marknadsanalysen, produkten visas med en systembeskrivning, projektutvecklingen med en processbeskrivning och näst sista presenteras erhållna erfarenheter. Utöver detta presenteras en översikt av samtliga individuella rapporter.

5.1 Marknadsanalys

För att ta fram krav till produkten Smart City utförde analysansvarig en marknadsanalys enligt de fem steg som tas upp i kapitel 4.1. Med hjälp av dessa steg kunde en mall metodiskt arbetas fram för hur kraven skulle se ut.

5.1.1 Steg 1

Under workshoppen i steg ett diskuterades saker som, vad produkten skulle göra, till vilka den ska vara riktad och hur den ska användas. När idén hur produkten skulle se ut tagits fram började analysansvarig med att leta information om liknande produkter enligt steg två.

5.1.2 Steg 2

I steg 2 hittades tre liknande produkter som redan finns på marknaden, se dem listade nedan.

Google Earth Outreach är en *Google Earth* implementation där olika geografiska verktyg framtagna av Google kan framställa unika kartor. Dessa kartor kan vara allt från personliga resor där dokumentation om sin rutt, till ekonomiska förändringar i en region. Några av de verktygen som används är *Google Street View*, *Open Data Kit* och *Geo Data Upload*. [28]

Event map - Emergency and Disaster Information Service är en färdig lösning från *The Hungarian National Association of Radio Distress-Signalling and Infocommunications (RSOE)* som går ut på att de tar in händelser från olika källor och länder. Där den mest informationen kommer från VMA-service (viktigt meddelande till allmänheten) från respektive land. [29]

Esri - Skapa Smarta Kartor är en helhetslösning som erbjuder API:er, databaser och analysering av data. Denna helhetslösning skulle göra det lättare för många att göra egna kartor och hålla den uppdaterad med relevant information. Lösningen erbjuder också möjligheten att ta in data i realtid och få datan analyserad på olika nivåer. [30]

5.1.3 Steg 3

För att särskilja Smart City på marknaden togs enligt projektgruppens åsikter fördelar och nackdelar fram från produkterna i steg 2, se punktlistorna nedan. Dessa fördelar och nackdelar behövde inte gälla alla produkter.

Fördelar

- API som tar in data för att sedan visualisera.
- Olika visualiseringsmöjligheter som passar den datan som du vill visa.
- Analys av data.
- Färdiga gränssnitt för att användare utan programmeringsvana ska kunna använda det.
- Molnbaserade.

Nackdelar

- Många är låsta till att bara visa tvådimensionellt.
- Inte många som tar emot *livedata*, utan arbetar med statistisk data.
- De färdiga lösningarna verkar lite osmidiga gällande användningen av data från olika källor.

5.1.4 Steg 4

Under workshoppen i steg 4 diskuterade gruppen fram sex egenskaper produkten skulle ha från steg 3 som kan ses nedan.

- En tredimensionell karta.
- Vill göra det lätt att ta in data från olika källor.
- Visualisera datan som händelser.
- Bygga ett modulärt system.
- Visualisera kartan i UE4.
- Bygga en egen databas som håller datan.

5.1.5 Steg 5

Kraven på produkten som arbetades fram kan läsas i tabell I.1.

5.2 Systemanatomi

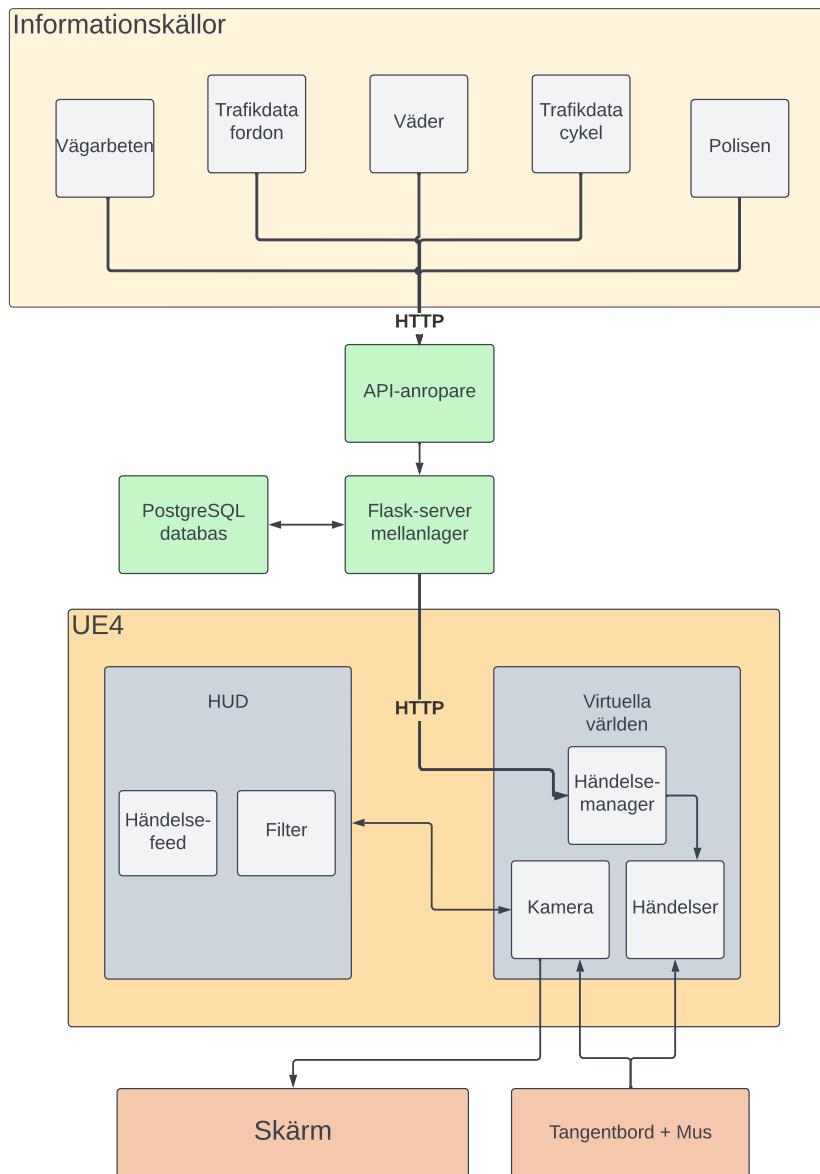
I början av projektet togs en systemanatomi och ett relationsschema för databasen fram, detta för att ge en övergripande bild över hur systemet skulle komma att se ut. Detta kände medlemmar i gruppen gav en överblick över hur systemets delar skulle samverka och hänga ihop. Detta tyckte medlemmarna ledde till att det blev lättare att planera utvecklingsfasen tack vare att det var enklare att se hur olika delar av systemet berodde på varandra innan det var utvecklat.

5.3 Systembeskrivning

För att lättare strukturera projektet har systemet delats upp i tre delsystem:

- UE4-klienten
- Projektets databas
- API-hanterare för datainsamling

Hur de tre delsystemen interagerar med varandra kan ses i figur 5.1. I UE4-klienten kommer Linköpings karta att presenteras i 3D där händelser kan ses. Ett användargränssnitt, se figur 5.2, kommer användas för att ge användaren information om händelsen. Användargränsnittet har funktionalitet för att låta användaren filtrera datakällor efter behov. Informationen hämtas från systemets databas som lagrar alla händelser. Händelser hämtas in till systemet genom att de olika datakällorna anropas efter ny information som sedan laddas upp till databasen. I figur 5.7 korresponderar 1, 2 och 3 mot b, a och c i figur 5.5. Nedan presenteras de tre delsystemet i närmare detalj.



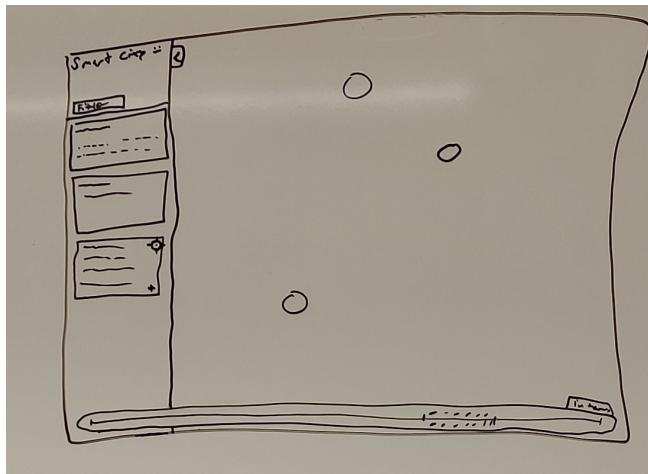
Figur 5.1: Systemöversikt

5.3.1 UE4-klient

UE4 driver visualiseringen av staden och den data som samlas in. I spelmotorn skapas grunden till världen med hjälp av Cesium. Världen består av en verklighetstrogen tredimensionell jordglob från Cesium som har förmågan att använda verklighetsbaserade koordinater. På jordgloben finns en karta med tredimensionella byggnader från Bings-kartor.

För att interagera med händelser och staden har ett användargränssnitt tagits fram. Se figur 5.5 för en ritad prototyp över hur användargränssnittet skulle se ut. Användaren kan åka runt i världen genom att styra en kamera placerad i världen. Ovanpå det kameran ser visas användargränssnittet, i detta kan händelser listas och filtreras efter ett antal söktermer.

För att strukturera och presentera utvecklingen av UE4, har ett diagram för hur komponenter kan tänkas sitta ihop tagits fram, detta visas i figur 5.6.



Figur 5.2: Bild från en workshop tidigt under utvecklingen där gruppen bestämde hur gränssnittet skulle se ut.

Händelser hämtas in från systemets databas in till UE4 genom HTTP-anrop. Detta görs genom en intern *händelsehanterare* som är placerad i den visualiseringen. Händelsehanteraren är en C++ klass skapad av projektgruppen som kommer agera som ett kontrolltorn för händelser. Händelsehanteraren hämtar information från databasen varje minut och placeras händelser i världen.

Kameran

Förflyttningen över kartan sker genom en anpassad kamera som kan styras med muspekaren eller tangentbordet. Den anpassade kameran skiljer sig från standardkameran i UE4 genom att begränsningar på hur den kan förflyttas. Till exempel kan användaren inte ändra höjden från marken, vilket begränsades eftersom användaren annars skulle kunna flyga igenom kartan.

Planen från början var att användaren bara skulle kunna flytta kameran med musen men eftersom det var enklare i början att programmera tangentbordet för kamerarörelser behölls de kontrollerna. Programmet börjar i musläge men kan gå över till tangentläge genom att trycka på mellanslag. Ovanpå kameran placeras användargränssnittet där händelser kan ses i en lista. Musen används för att interagera med användargränssnittet i båda lägen.

Gränssnitt

Gränssnittet sköter logiken för uppvisning av *widgets*, de visuella objekt som används för att visa upp händelser och generell information för användaren. Med dessa widgets byggs visuella lager upp och det blir lättare att instansiera objekt av dessa typer med hjälp av den logiken som finns i C++-koden för dessa widgets. Varje lager sköter sin egen logik och funktioner vilka anropas med hjälp av UE4 *blueprints*, de sköter kopplingen mellan dessa lager och allt annat sköts med C++-kod.

Visualisera händelser

Händelser visas på kartan genom utplacerade koner. Användaren kan interagera med händelser genom att klicka på dem i världen varpå händelsens information visas upp i en textruta. För att minska antalet händelser som visas kan användaren filtrera bort händelser med avseende på deras typ. Användaren kan då till exempel välja att endast se cykelrelaterade händelser varpå endast de händelserna placeras ut i världen. Förutom att klicka på händel-

ser kan deras information även visas i användargränssnittet. Till vänster av skärmen kommer ett flöde visa de händelser som är utplacerade och dess information.

Händelsehanterare

När händelser hämtas från databasen skickas de i JSON-format till händelsehanteraren. Händelsehanteraren behandlar datan genom att hämta ut och skapa händelseobjekt från varje JSON-objekt. Händelseobjektet lagrar händelsens information i sju attributer som matchar databasens lagring av information.

Händelseobjekt

För att lagra händelsers information skapades en händelseklass där datan läggs in i klassens fält.

Informationen kan visas för användaren genom kartan där händelser är placerade. Händelsens information består av dess id, datum, stad, titel, generell information, typ och koordinater. I händelsens *EventInfo* sparas den specifika informationen för händelsen. Till exempel kommer antalet cyklar för typen cykelinformation sparas i info och vädret kommer sparas för typen väder. Informationen i *EventInfo* är alltså unik för typen av händelsen, se figur 5.3 nedan för två exempel på händelser i dess JSON-format.

```
{
  "7": {
    "city": "Linkoping",
    "date": "Mon, 09 Mar 2020 00:00:00 GMT",
    "info": "Roadwork from 2020-03-09 to 2022-06-03",
    "latitud": 15.622143617421179,
    "longitud": 58.41622284247436,
    "title": "Sturegatan 3a",
    "type": "Roadwork"
  },
  "8": {
    "city": "Linkoping",
    "date": "Thu, 25 Jun 2020 00:00:00 GMT",
    "info": "Roadwork from 2020-06-25 to 2022-05-31",
    "latitud": 15.524333643613462,
    "longitud": 58.4842112346546,
    "title": "Stjornorpsvagen 3",
    "type": "Roadwork"
  }
}
```

Figur 5.3: Exempel på svar från server

5.3.2 Datainsamling

Datainsamlingen innefattar den del av systemet som samverkar med externa databaser för att utvinna information. Detta genomförs via HTTP-protokoll där en modul för anrop till varje informationskälla skrivits. De informationskällor som hittills tagits in i systemet kan ses nedan:

- Linköpings kommun:
 - Vägarbeten [31]
 - Trafikdata fordon [32]
 - Trafikdata cykel [33]

-
- Polisen [34]
 - SMHI: Väder [35]

När en modul hämtar ny data kommer den att tillhandahålla databasen ny data direkt. För att systemet ska kunna hantera att automatiskt hämta ny information har ett skript skrivits för att starta inhämtningen. Varje modul har information om hur ofta källorna uppdateras med ny data för att göra automatiska anrop när ny data finns.

Vissa datakällor har begränsningar på antal anrop som får göras under en viss tid, vissa har andra begränsningar som att anrop tar flera sekunder, exempelvis Linköpings kommun som tar i snitt 20 sekunder per anrop. Detta betyder att systemet inte kommer kunna presentera informationen i realtid utan är beroende av hur lång tid anrop tar och hur ofta det är möjligt att anropa källan.

När anrop gjorts till datakällorna formateras datan om för att passa databasen, detta kan exempelvis vara att formatet på datum ska ändras eller att koordinater behöver omvandlas. Flera datakällor formaterar data på olika sätt, därför behövde systemet skrivas i olika moduler för att hantera detta.

5.3.3 API

Systemets API är skrivet i *Python* med ramverket *Flask*. API:ets huvuduppgift är att underlätta kommunikationen från datainsamlingen och UE4-klienten till databasen. Datainsamlingen använder det för att lägga upp ny data på databasen och UE4-klienten använder det för att hämta nya händelser från databasen.

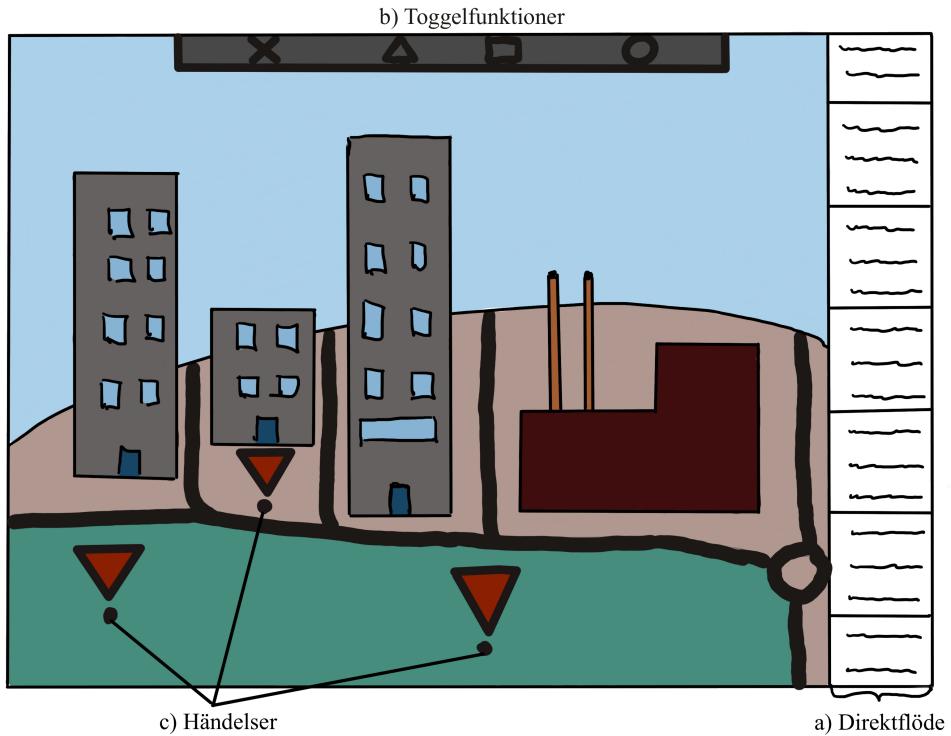
API:et anslutar till databasen med biblioteket *psycopg2*, när den har anslutit till en databas använder utvecklaren den för att skriva *SQL* förfrågningar till databasen. De förfrågningarna används både för att skapa och hämta data.

5.3.4 Databas

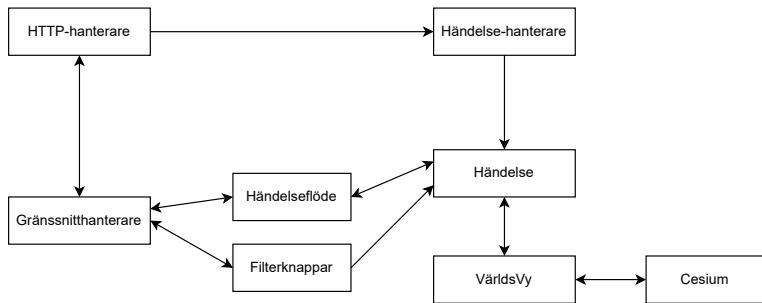
Som databas används PostgreSQL, denna lagrar all information som utvinns från informationskällorna. Strukturen på denna databas kan ses i 5.4.

Event							
id	date	city	title	info	type	longitude	latitude

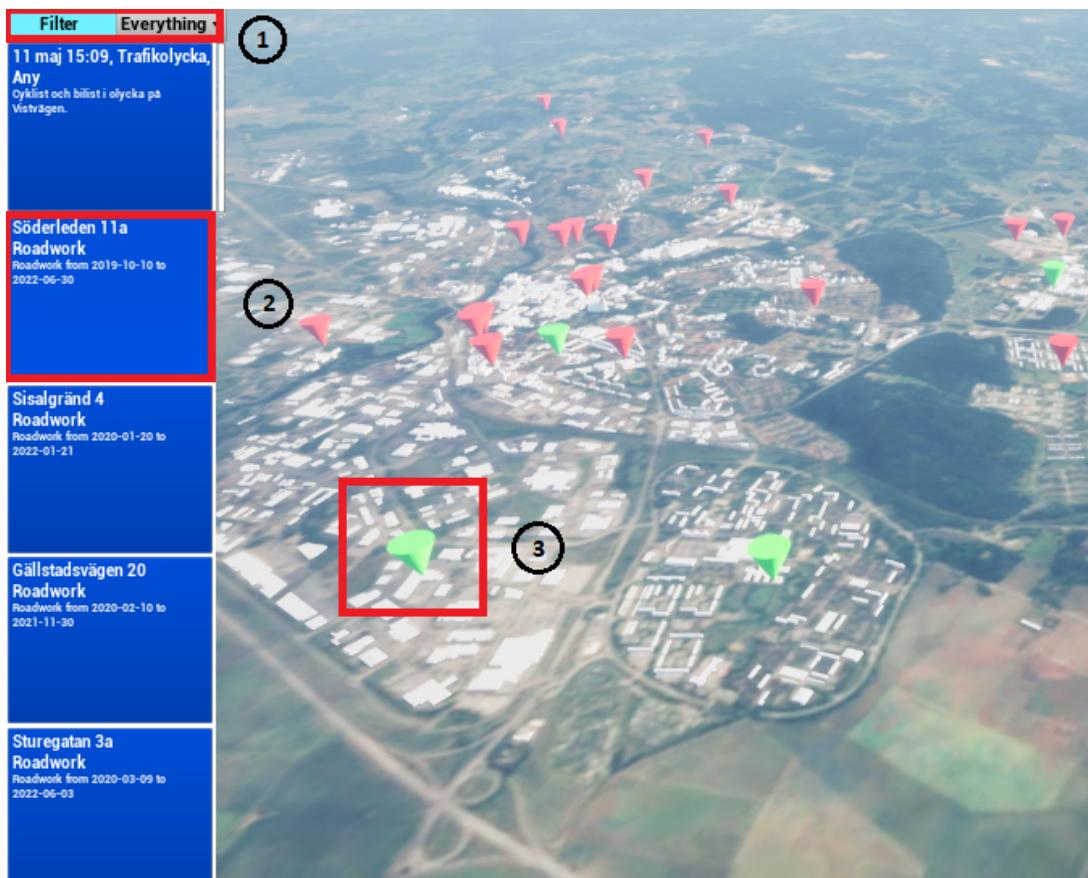
Figur 5.4: *Databasstruktur*



Figur 5.5: Systemets visualisering gränssnitt.



Figur 5.6: Systemets arkitektur i Unreal Engine 4



Figur 5.7: Bild från när systemet körs och allt som implementerats fungerar.

5.4 Produktutvärdering

Produkten uppfyller 73% av de krav med prioritet 1. Totalt uppnåddes 67% av kraven, se tabell I.1 för en sammanställning av samtliga krav.

5.5 Processbeskrivning

I detta kapitel beskrivs resultatet av de processer som gruppen använde för att ta sig framåt i arbetet.

5.5.1 Arbetsprocess

Som arbetsprocess användes delar från Scrum-metodiken och kanban-tavlor i Trello i kombination med utvärderingsmöten för att effektivisera processen. Mer om arbetsmetodikerna går att läsa om i avsnitt 4.2.1 och 4.2.2.

Svaren från enkäterna sparades och diskuterades under varje utvärderingsmöte för att ge inblick på ifall arbetsprocessen upplevdes som effektiv av gruppmedlemmarna. Sammanställda svar från enkäterna kan ses i tabell 5.1 där sju är högsta poäng och ett är lägst.

Fråga	Sprint 3	Sprint 4	Sprint 5	Sprint 6
Hur väl nåddes de mål som sats upp under denna sprint?	3,75	4,35	5,25	4,80
Hur känner du att din insats i projektet varit denna sprint?	3,75	5,38	5,13	5,60
Hur upplevdes senaste sprinten?	3,50	4,63	5,63	4,20
Kände du att tiden räckte till?	2,88	4,25	5,50	4,00

Tabell 5.1: Medelbetyg för varje fråga för varje sprint.

När Scrum används brukar det finnas förvalda aktiviteter som gruppen väljer eller tilldelas att arbeta på. Däremot märktes det under projektet att det var svårt att hitta förutbestämda och konkreta aktiviteter, särskilt till UE4. Detta berodde dels på att gruppmedlemmarna hade bristande erfarenhet av att utveckla i UE4 och att det därför var svårt att dela upp uppgifter i mindre delar. Därför beslutade gruppen att varje person skulle sätta upp arbetsuppgifter inom det område de jobbade med. Hur gruppen kom fram till detta var från svaren som dokumenterades i enkäterna och lösningen kom fram genom diskussion under utvärderingsmötena.

En annan ändring som gjordes under projektet var att korta ner längden på sprinten. Varje sprint var i början två veckor långa men blev från och med sprint 4 en vecka långa. Detta togs upp i ett av utvärderingsmötena eftersom flera i gruppen ansåg att varje sprint var för lång. Den initiala längden på sprinten ledde till att gruppmedlemmarna hann glömma bort vad som skulle arbetas under sprinten. Genom att korta ner längden på sprinten fick gruppen en bättre blick på vad som förväntades göra under den kommande veckan.

I samband med att gruppen bytte till en-veckas sprinter infördes enkäter för att samla in erfarenheter. Därväg implementerades enkäterna först vid sprint 3.

Ytterligare en förändring togs upp i sprint 3. Flera i gruppen kände att de inte hade koll på vad andra i gruppen höll på med under dagen. För att motverka detta skapades en kanal på Discord där gruppmedlemmarna i början av varje dag skulle ge en kort sammanfattning på vad de tänker arbeta med under dagen. Effekten av detta är dock svår att avgöra från enkäterna.

5.6 Mätning av kvalitetskrav

Projektet hade totalt tre kvalitetskrav som skulle mätas. Tanken med dessa var att under projektets gång se hur dessa kvaliteter uppehålls. Kraven som mättes var:

1. All Pythonkod måste följa formateringsstandarderna för PEP 8.
2. All kod ska klara alla tester i pipelinen på Gitlab.
3. 80 % av alla Python-funktioner ska ha docstrings.

Det första kravet uppnåddes genom att sätta ett villkor med en pipeline på att all kod måste ha blivit formaterad av programmet Black, som kan formatera kod till PEP 8. På samma sätt sattes kravet på att all Python-kod ska klara alla tester. Eftersom inget program som hjälpte med det tredje kravet hittades i tid beslutades det att alla funktioner skulle läsas manuellt för att säkerställa vilka funktioner som har docstring.

5.7 Gemensamma erfarenheter

I detta kapitel tar projektgruppen upp de erfarenheter som sammanfattades från erfarenhetsmötet i kapitel 4.6.4.

5.7.1 Kvalitetscoachning

Under projektetgång hade gruppen workshoppar med kvalitetsstudenterna. På dessa möten lyftes de viktiga delarna som behövde finnas med i kvalitetsplanen och det allmänna projektarbetet. Detta resulterade att gruppen fick en bättre bild av hur kvalitetsplanen skulle kunna se ut och hur den skulle följas upp under projektets gång.

5.7.2 Kommunikation

Under projektet hade gruppen två plattformar för intern kommunikation. Discord användes som den primära kanalen för utvecklingsrelaterad kommunikation. Här har gruppmedlemmar strukturerat upp med specifika kanaler för varje område, till exempel test-, kvalitet- och arkitektspecifik kommunikation. Det har även varit den plattform gruppen använt vid parprogrammering på distans eller andra möten på distans. Det har även varit där uppdateringar om hur projektets status utvecklats när gruppen inte arbetat tillsammans fysiskt. För mer allmän och vardaglig kommunikation har gruppen använt Facebook Messenger, där kortare uppdateringar om exempelvis kommande möten skickats ut till gruppen. Att dela upp kommunikationen på det här sättet kändes naturligt och framstod som att det hjälpte kommunikationen i arbetet.

Projektgruppen har även haft en gemensam kalender där möten och inplanerade, gemensamma arbetspass lades in. Detta gjorde att gruppen fick bättre koll på hur mycket schemalagd tid som fanns och hur mycket tid som fanns kvar till självstudier som rapportskrivning.

Gruppen försökte skapa god kommunikation under projektets gång. Ett sätt att förbättra den var genom att gruppen hade ett flertal träffar utanför arbetsstider under projektets gång. Genom dessa träffar lärde gruppen känna varandra relativt fort. Att vara tillsammans även utanför projektarbetet kändes kommunikationen i gruppen mer naturlig och avslappnad.

5.7.3 Möten

Gruppen hade regelbundna möten i början av varje vecka där den kommande veckan planerades upp och viktiga händelser eller deadlines togs upp, dessa möten ledde också till att alla i gruppen fick möjlighet att ta upp eventuella frågor eller funderingar som dykt upp. Tidigt i projektet upplevdes dessa möten lite ostrukturerade, vilket berodde på att gruppen i början inte hade god koll på vad som faktiskt behövde göras i projektet. Allt eftersom att projektet utvecklades och mognade blev mötena mer konstruktiva och kom att innehålla moment där varje gruppmedlem fick berätta vad de åstadkommit under veckan och vad som saknas för att en modul ska vara färdigställd.

5.7.4 Unreal Engine 4

Under en längre tid var utvecklingen av UE4-klienten stagnerad då motorn inte fanns tillgänglig på datorerna på campus. Detta ledde till att gruppen inte kunde utveckla klienten tillsammans på plats och ingen i gruppen hade en bärbar dator kapabel att starta mjukvaran. Det blev då svårt för en del av gruppen att vara delaktiga i utvecklingen som skedde i UE4. Delar av gruppen ägnade sig istället åt parprogrammering på distans för att utveckla i UE4. När mjukvaran väl blev tillgänglig på plats var det svårt för vissa i gruppen att komma in i utvecklingen. De som dock hade erfarenhet av utveckling av mjukvaran flyttade utvecklingen över till att vara på plats vilket resulterade i att framtagandet av en första prototyp gick snabbare.

5.7.5 Övergången till marknadsanalys

En annan sak som skedde under projektet var att gruppen av särskilda skäl valde att bryta samarbetet med den tänkta kunden, som skedde ungefär halvvägs genom projektet. Detta ledde till att gruppen kunde arbeta mer självständigt, inte var beroende av att kommunikationen med kunden fungerade och istället kunde ta egna beslut baserade på undersökningar gruppen själva genomförde. En konsekvens var dock att gruppen var tvungen att göra om hela kravanalysen som från början grundades i kundens krav och istället behövde göra en kravanalys baserad på en marknadsanalys. Detta var en tidskrävande process men det ledde till att gruppen istället kunde utveckla en produkt de ansåg hade marknadsvärde och kunde utveckla med verktyg de ansåg vara lämpade för produkten.

5.7.6 Utvärdering av arbetsmetoder

Arbetsmetoden som valdes upplevdes till en början som luddig av en del av gruppmedlemarna. En orsak till detta var att gruppen, inklusive gruppens Scrum-mästare, hade dålig koll på hur arbetsmetoden skulle appliceras i praktiken. I början valde gruppen att använda sig av två veckors-sprint, vilket ledde till att gruppen inte kände att det fanns några tydliga interna deadlines eller kortsliktiga mål att jobba mot. Efter cirka halva projekttiden bestämde gruppen för att ändra sprint-längden till en vecka och började genomföra mer strukturerade utvärderingar. Detta gav en förbättring i arbetet eftersom målen som sattes upp inför varje sprint diskuterades mer flitigt och gruppen fann mer klarhet i vad som hade gjorts och vad som kvarstod att göra.

Eftersom det inte fanns en möjlighet att sitta på plats och jobba med vad som rörde UE4 blev det ibland otydligt vem som jobbade med vad och hur långt personen hade kommit. Det gjorde det även extra svårt för gruppmedlemmar att komma in i den redan påbörjade utveckling. När gruppen väl fick möjlighet att jobba tillsammans på campus upplevdes arbetet som mer strukturerat och effektivt. Det hade varit positivt om möjligheten att jobba på campus tillsammans funnits tidigare i projektet, men gruppen hade ingen direkt möjlighet att påverka detta.

En till anledning till att gruppen inte hade full koll på vad som arbetades på var att gruppmedlemmar inte uppdaterade eller använde den Kanban-tavla som fanns för varje sprint. Hade Kanban-tavlorna använts som planerat och hade mer detaljerad information hade detta problem nog inte uppstått i samma utsträckning.

5.7.7 Roller

I början av projektet tilldelades varje person en unik roll med ett eller flera ansvarsområden, vilket ledde till att arbetet kunde delas upp mer naturligt när alla redan hade ett ansvarsområde. Detta gjorde också att gruppen hade en form av expert eller ansvarig för varje område inom projektet, vilket gjorde att resten av gruppmedlemmarna inte behövde ha samma koll på alla delar av projektet. Det ledde också till att gruppmedlemmarna visste vem som skulle kontaktas vid specifika frågor, både gällande utveckling och arbetsmetod.

Gruppen tycker att rollerna hade stor påverkan i projektets planeringsfas när exempelvis en arkitektur, struktur för Git och arbetsmetoden Scrum skulle tas fram. När gruppen bröt samarbetet med kund ändrades ansvarsområdena för analysansvarig och allt ansvar rörande kundkontakt försvann. Detta gjorde att analysansvarig fick ett mindre ansvarsområde och i stora drag inte hade något eget ansvarsområde när marknads- och kravanalysen var färdigställd. I efterhand kunde gruppen konstatera att det hade varit bra att lägga till en specialistroll som hade ansvar över hur UE4 användes, denna roll hade i detta fall analysansvarig kunnat ha som en extraroll. Detta hade antaligen resulterat i en mer strukturerad utveckling

i UE4 än den som blev. En stor anledning att utvecklingen i UE4 blev ostrukturerad tros vara konsekvens av att ingen hade tidigare erfarenhet av verktyget.

En roll som användes oväntat mycket var dokumentansvarig eftersom det var många dokument som behövde skrivas för att fullfölja projektet. Det var väldigt fördelaktigt och tidsbe-sparande för gruppen att det fanns en person som ansvarade för att samtliga dokument följe en internt bestämd standard.

5.8 Översikt över individuella bidrag

Till detta projekt ingick att varje gruppmedlem skulle skriva en individuell rapport som bila-ga till denna rapport. Syftet var att gräva djupare i aspekter kring detta projekt och ge djupare förståelse kring dessa. Nedan listas alla gruppmedlemmar med deras individuella bidrag.

5.8.1 Maya Henriksson Björklund – Att agera på aggregerad data

En analys av de samhällsrisker som finns kopplade till Smart City och aggregerad data, rap-porten beskriver metoder för att utöva risk- och sårbarhetsanalyser, hur terrorhotnivån för dagens samhälle ser ut och vilka risker som finns kopplade till att kombinera större mängder data.

5.8.2 Filip Josefsson – Unreal Engine: Mer än bara underhållning

En fallstudie med syftet att upptäcka hur Unreal Engine har använts utanför underhållnings-syften. Rapporten diskuterar om vad som gör att spelmotorn lämpar sig i dessa fall och kon-sekvenserna av att använda motorn. Huvudfallen som upptäcktes var att spelmotorn fram-förallt användes till utbildning och simulerings.

5.8.3 Daniel Kouznetsov – Spelmotorers påverkan på designprocessen

En litteraturstudie om vad för påverkan spelmotorer och Virtual Reality kan ha på designpro-cesser, hur VR kan användas för att skapa en mer interaktiv designprocess och en diskussion om vilka branscher som har användning av detta.

5.8.4 Tony Lindbom – Metodik vid framtagning av krav

En litteraturstudie om vad det finns för metoder för att ta fram krav för ett mindre projekt och vad som kan ge värde för en kund.

Denna studie resulterade i flera olika metoder och hur de kan användas för att ta fram krav till en kravspecifikation och vad de har för olika för- och nackdelar.

Det diskuteras också i rapporten om vad som skulle vara ett bättre alternativ än den metod som användes för *Smart City*

5.8.5 Simon Magnusson – Hur uppmärksammar man en användare när det finns oändligt med information

En litteraturstudie av faktorer som kan påverka en användares effektivitet och uppmärksam-het i ett system. Texten fokuserar på hur man kan minimera den kognitiva ansträngning som olika gränssnitt kan ha på användare.

5.8.6 Viktor Norgren – Hur kan Q-learning användas för att testa ett användargränsnitt utvecklat i Unreal Engine 4

En litteraturstudie och en implementation kring att hitta en metod som använder *Q-learning* för att automatiskt testa Smart City:s användargränssnitt.

5.8.7 Gustav Peterberg – Analys av flaskhalsar i systemet Smart City

Denna rapport ämnar att hitta analysmetoder som lokalisera flaskhalsar och presentera metoderna lättförståeligt. Metoderna kan då tänkas implementeras av framtida utvecklare i till exempel Smart City men även liknande eller större mjukvarusystem. Studien resulterade i två metoder för att lokalisera ett systems flaskhalsar. En generell metod som är lättare att implementera och FOREPOST, en mer grundlig men svårare metod för att hitta flaskhalsar. I rapporten diskuteras fördelen av att använda autonoma tester jämfört med manuella tester, de potentiella effekterna av att lösa flaskhalsarna. Men även förbättringsmöjligheter lyfts för att diskutera alternativa implementationer.

5.8.8 Anna Zhao – Versionshantering med Git för Unreal Engine 4

En studie och diskussion om unika aspekter när det gäller versionhantering av projekt skapad i spelmotorer och hur det kan hanteras, där versionshanteringsverktyget Git och spelmotoren UE4 undersöks närmre.

Det resultat rapporten kom fram till var att unika aspekter med att versionhantera projekt skapade i spelmotorer som UE4 är att de genererar binärfiler under utvecklingen som kan vara svåra att versionhantera. Detta på grund av att binärfiler både är näst intill omöjliga att *merga* och att de kan ta upp stora mängder data både när de skapas och när de versionhanteras.

Rapporten kom även fram till att UE4 har stöd för Git som versionshanteringsverktyg och att det finns verktyg för Git som ger alternativ för att hantera binärfiler. Det finns till exempel *Git LFS* som kan underlätta versionhantering av stora filer genom en extern server. Både *Git LFS* och *GitLab* har funktionen att låsa filer vilket är ett alternativ till att hantera binärfiler på ett sätt så att man minskar chansen för *merge* problem.



6 Diskussion

Detta kapitel tolkar resultatet och utvärderar projektets arbetssätt och metoder.

6.1 Resultat

Detta kapitel diskuterar projektets resultat och innehåll, vad som hade kunnat göras annorlunda och lärdomar projektetsmedlemmarna tar med sig.

6.1.1 Implementationsval och implementationsalternativ

Den mest tidskrävande delen av projektet var att utveckla UE4-kartan. Detta berodde främst på att gruppen inte hade tidigare erfarenhet med UE4. Gruppen fick bland annat lära sig hur Blueprints fungerade, hur C++-kod kombineras med Blueprints och fick även lära sig svårigheterna med versionshantering i UE4. Anledningen till att UE4 användes i projektet var att kunna nyttja Cesium då detta verktyg underlättade för visualisering av kartan. En annan anledning att gruppen valde just UE4 var för att den initiala kunden hade detta som önskemål. Trots avbrutet samarbete med kund valde gruppen ändå att stanna vid den planen, detta då gruppen inte såg något större syfte med att byta utvecklingsmiljö. UE4 tillhandahöll de funktionen projektet krävde.

UE4-miljön bestod i helhet av tre delar som för det mesta kunde utvecklas separat från varandra. Gruppen kunde fokusera på att få varje individuell del färdig från början. Den uppdelade arbetsstrategin gynnade gruppen när det kom till att lära sig om hur motorn fungerar men det resulterade i att det tog lång tid att ta fram en riktig prototyp.

Att gruppen valde UE4 som utvecklingsmiljö för den visuella delen trots att ingen hade tidigare erfarenhet gjorde att gruppen i övrigt endast använde sig av verktyg det fanns tidigare erfarenheter av. Exempelvis hade InfluxDB möjligtvis varit en mer lämpad databas för att hantera händelser med kontinuerlig data. För att göra ett exempel skulle cykeldatan från Linköpings kommun hamna i den databasen. Detta beror på att i InfluxDB skulle cykel-datans händelse kunna uppdateras kontinuerligt, tillskillnad från den nuvarande implementation där en ny händelse läggs till i databas varje gång datan uppdateras. Anledning till att denna lösningen inte användes var för att projektgruppen trodde att det skulle ta för lång tid att utveckla två databaser, istället för en.

Anledningen till att gruppen valde att använda Flask var för att medlemmar i gruppen hade god erfarenhet av verktyget sedan tidigare. Innan samarbetet med kunden avslutades var tanken att använda *Swagger*. Detta alternativ valdes dock bort då ingen hade erfarenhet av *Swagger* sedan tidigare, därfor kände gruppen att det var säkrare att välja Flask.

6.1.2 Systemarkitektur

Ett av de beslut som diskuterades inom gruppen var huruvida UE4-miljön skulle kommunicera med databasen direkt eller genom ett mellanlager som skötte den kommunikationen. Att interagera med databasen direkt från UE4 har den tydliga fördelen att det hade varit snabbt, hade låtit Flask-servern bli helt utformat med ändamålet att lägga in nya händelser i databasen från API-anropen. Gruppen valde dock att sköta kommunikationen mellan databasen och UE4 genom ett mellanlager, anledningen var att direkt kommunikation hade gjort det svårare att vidareutveckla systemet utan att störa UE4-klienten. Att ha ett mellanlager gjorde det möjligt att ändra strukturen på databasen utan att behöva ändra direkt i UE4-klienten, vilket gör systemet mer modulärt.

6.1.3 Hur kan produktens värde på marknaden ökas?

Resultatet av projektet blev ett system som visar grundidéns potential. För att bygga vidare på produkten finns det vissa funktioner som kan tänkas vara relevanta att utöka systemet med för att göra det mer användbart.

Fler datakällor

En utökning är att implementera fler informationskällor för att fylla på kartan med mer händelser för att kunna ge användare mer information att utgå från. Detta ökar användningsområdet och därmed antal personer som kan vara intresserade att använda produkten.

Att göra det lättare för användaren att implementera egna datakällor hade varit en stor förbättringspunkt för produkten, hur detta skulle genomföras kom projektgruppen dock aldrig fram till. Detta hade gjort att användaren själv kan lägga till egna källor och därmed skräddarsy produkten helt efter egna behov.

Tidslinje

En tidslinje är en funktion som skulle gynna produkten. Med fler händelser att utgå ifrån skulle en funktion som låter användaren tydligare se ett händelseförlopp skapa värde för produkten. Detta skulle kunna skapa möjligheten till att se samband för olika typer av händelseförlopp som inte vore lika tydliga utan. Exempel kan vara att visualisera om det finns någon korrelation med exempelvis polisuttryckningar och mängden folk i ett område. En sådan funktion kan ge värde för att det skulle ta produkten från något som endast representerar reallidsinformation till något som gör det möjligt att granska händelseförlopp visuellt.

Maskininlärning

Med en tidslinje kan användaren blicka tillbaka i tiden och få en översikt av vad som hänt. För att blicka framåt i tiden kan maskininlärning implementeras på den data som Smart City har tillgänglig. Genom att välja lämpliga algoritmer kan framtida händelser möjligens förutspås och vara till hjälp för framtida polisuttryckningar eller larmuttryckningar.

6.2 Metod

I detta kapitel diskuteras de val av arbetsmetoder gruppen arbetade med.

6.2.1 Scrum

Projektgruppen valde tidigt i projektet att arbeta enligt en kombination av Scrum och Kanban. Detta eftersom det ansågs passande för projektet och för att skaffa erfarenhet av att använda dem i praktiken. Något som märktes under projektet var att två veckors sprinter var för långa för utveckling av produkten. Gruppen kände att arbetet blev mindre strukturerat och kommunikationen om vem som gjorde vad var bristande. En av dessa anledningar var för att gruppens kanban-tavla inte användes som planerat. Detta ledde till medlemmarna gruppen inte visste vad som jobbades på vid tillfället och vem som hade planerat att göra vad. Hade kanban-tavlans tillsammans med fler möten under sprintet hade en sprint-längd på två veckor säkert varit framgångsrika.

Projektgruppen valde att byta sprintlängden till en vecka kort in i utvecklingsfasen. Detta gjorde att projektgruppen mer naturligt hade tätare kontakt och avstämning gällande utvecklingsstatus. Detta då det exempelvis gjordes sprint-enkäter oftare vilket ledde till att projektgruppen fick bättre koll på hur medlemmarna upplevde projektet. Gruppen märkte en klar förbättring av att använda enkäterna. De förbättrade arbetsprocessen genom att gruppen kunde korrigera utifrån enkätens svar och då förbättra kommande sprint.

6.2.2 Kanban

Projektgruppen hade svårt att hålla kanban-tavlans kontinuerligt uppdaterad under samtliga sprinter. En anledning till detta kan ha varit att medlemmarna inte var vana vid att jobba med en kanban-tavla och att det därmed inte var naturligt att fylla i den. En annan anledning kan ha varit att medlemmarna jobbade ganska uppdelat och därmed inte kände att det fanns något större syfte med att uppdatera aktiviteterna på tavlan då det var ett fåtal personer som jobbade med den aktiviteten.

Att kanban-tavlans tillsammans med fler möten under sprintet hade en sprint-längd på två veckor ledde bland annat till att det blev svårt för gruppmedlemmar att hoppa in i utvecklingen då det inte var uppenbart vad som jobbades på och vad som var kvar. Hade en kanban-tavla använts som planerat hade arbetet antagligen blivit mer strukturerat och effektivt.

6.3 Arbetet i ett vidare sammanhang

Detta kapitel diskuterar projektet med avseende på etik, samhälle och miljö.

6.3.1 Etiskt ansvar

Smart City kan bidra till att invånare känner att de lever i en form av övervakningssamhälle där allt som sker dokumenteras. Trots att Smart City inte skapar egna händelser kan ihopräkningen av data göra att användare blir mer medvetna om hur mycket information som faktiskt finns om både samhället och individer.

Det finns en risk till övertrö på den informationen som Smart City innehåller. Då Smart City inte kontrollerar den data som hämtas in från källorna finns det inget sätt att garantera att den faktiskt stämmer. Hade produkten exempelvis använts av en stadsplanerare som analyserat cykeltrafiken med hjälp av Smart City kan detta till exempel leda till felaktiga beslut tas ifall informationen som visas inte stämmer. Om Smart City ofta visar felaktig information kan det leda till att användare inte litar på produkten, vilket kommer leda till att den tappar

sitt värde. Att kontrollera om informationen som Smart City får in från externa datakällor stämmer är i princip omöjligt.

Smart City samlar in och aggregerar samhällsinformation från öppna källor, eventuella problem med detta beskrivs i kapitel A.

6.3.2 Samhällaspekter

Visionen med produkten är att den ska kunna användas av en larmoperatör för exempelvis SOS Alarm. I detta sammanhang kan då produkten användas för att larmcentralen ska kunna få en överblick av staden för att kunna göra val gällande utryckningar. Detta kan leda till att det blir lättare att skicka ut förebyggande hjälp, så som polispatruller om det samlats ovanligt mycket folk på en plats, vilket kan leda till att olyckor kan förhindras och förebyggas och i sin tur kan leda till att resurser kan besparas.

Om systemet skulle hämta data från sociala medier, exempelvis Twitter, skulle det vara fullt möjligt för användare att sabotera systemet genom att exempelvis medvetet lägga upp felaktig information om händelser. Ifall kartan av någon anledning skulle visa felaktig information kan det dock ha motsatta konsekvenser, som istället kan leda till att resurser skickas ut i onödan eller att resurser tilldelas fel i förebyggande syfte.

6.3.3 Miljö och hållbar utveckling

För att produkten är tänkt att visualisera händelser i realtid kan det, med ytterligare datakällor, användas för att följa exempelvis strömvabrott, oljeläckage eller strömanvändning i realtid. Detta, tillsammans med annan data, kan tänkas vara användbart för att se korrelationer mellan dessa händelser och annat som sker i staden. Går dessa korrelationer att dra kan det eventuellt göra det lättare att lägga fram en plan för att förhindra att exempelvis oljeläckage sker eller att göra det möjligt att minska elförbrukning.

Smart City kan användas vid stadsplanering för att exempelvis se vilka cykelvägar som är mest använda och planera staden efter detta för att effektivisera resvägar.



7 Slutsatser

Detta kapitel ämnar sig att svara på de frågeställningar som ställdes i början av rapporten.

7.1 Vilka funktioner i Smart City kan skapa värde för marknaden?

Enligt marknadsanalysen som utfördes hittades sex stycken funktioner som skulle särskilja Smart City, se avsnitt 5.1. Av dessa implementerades tre stycken, se punktlistan nedan.

- Systemet visualiseringen kartan i 3D.
- Händelser kan placeras och interageras med på kartan.
- Det är lätt att lägga till fler datakällor i systemet.

Smart City visualiseringen händelser i UE4 genom att presentera dem som koner av olika färger beroende på typ på en tredimensionell karta. Användaren kan sortera händelser med hjälp av ett filter som då presenterar ett flöde av de sorterade händelserna. Smart City är designad för att visualisera alla händelser som lagras i systemets databas enligt formatet i figur 5.4. Detta gör det enkelt att lägga till informationskällor.

7.2 Vilka erfarenheter kan dokumenteras från detta projekt och kan vara intressanta för framtida projekt?

De erfarenheter som kan dokumenteras från detta projekt är att god och genomgående kommunikation är grundläggande för att genomföra ett projekt i grupp. Det är fördelaktigt att tidigt i projektet bestämma hur arbetet ska gå till. Vidare är det viktigt att arbeta för att följa den bestämda processen med möjligheter för gruppen att gemensamt ta beslut om hur processen kan förbättras. Projektgruppen har under projektet fått kunskaper inom utveckling i UE4. Detta innefattar ökad kunskap inom C++ och Blueprints. Utöver detta fick gruppen även kunskaper av att använda GitLab för versionhantering av UE4 och andra delar av projektet.

7.3. Vilka utvecklingsmöjligheter finns till Smart City som skulle öka marknadsvärdet?

Under projektet fick projektgruppen erfarenheter av att arbeta enligt den agila arbetsmetoden Scrum tillsammans med Kanban. Gruppen fick erfarenhet av att ta fram en produkt utefter en marknadsanalys som kan läsas mer om i 4.1 och 5.1.

7.3 Vilka utvecklingsmöjligheter finns till Smart City som skulle öka marknadsvärdet?

Att göra det möjligt för användaren att själv lägga till datakällor på ett enkelt sätt hade ökat marknadsvärdet. Att även göra det möjligt att se en tidslinje av alla händelser hade ökat användningsområdet och därmed marknadsvärdet. För att även se fram i tiden skulle händelser kanske kunna förutspås med maskinell lärande utifrån tidigare händelser.

7.4 Vad för värde skapar en systemanatomi för ett projekt i denna storlek?

En systemanatomi kan underlätta för att ge gruppen en enhetlig bild av det system som ska utvecklas. Det kan även göra det lättare att se beroenden i ett system innan utveckling startar, vilket kan göra det lättare att utveckla saker i rätt ordning och dela upp arbetet.



A

Att agera på aggregerad data - Maya Henriksson Björklund

A.1 Introduktion

Smart City är ett system som samlar och visualiseras stora mängder information om samhället. Dessa händelser är allt från antal cyklar som passerat till polisuttryckningar. All denna data kommer från öppna källor och är alltså redan tillgänglig för vem som helst att ta del av men finns det problem med att samla all denna data på ett ställe? Att göra samhällsinformation lättillgänglig och överskådlig gör den lättare för människan att hantera och bearbeta, vilket öppnar för nya möjligheter men vilka risker följer med?

A.1.1 Syfte

Denna rapport har som syfte att reda ut de risker som aggregerad data och Smart City bidrar med på både individ- och samhällsnivå.

A.1.2 Frågeställningar

1. Vilka metoder finns för att analysera samhällsrisker gällande aggregerad data?
2. Finns det scenarion där Smart City tillsammans med andra produkter, i detta fall Google maps, som samlar liknande information ökar risken för oavsiktlig användning av antagonistiska grupper?

A.2 Teori

Detta kapitel tar upp den teori som är nödvändig för att ta del av rapporten.

A.2.1 Definitioner och akronymer

Aggregerad data

Att aggregera data innebär att samlar data, ofta från flera källor, för att sedan sammanställa denna data [36].

Totalförsvarets forskningsinstitut, FOI

Svensk myndighet som arbetar med forskning, teknik- och metodutveckling. FOIs största beställare är Myndigheten för samhällsskydd och beredskap samt Försvarsmakten. [37]

Myndigheten för samhällsskydd och beredskap, MSB

Svensk myndighet med ansvar för att stödja samhällets beredskap för olyckor, kriser och civilt försvar [38].

Säkerheitspolisen, Säpo

Svensk myndighet med nationellt uppdrag att bland annat förebygga och avslöja brott mot landets säkerhet och bekämpa terrorism [39].

Risk- och sårbarhetsanalys, RSA

En analys av risker och sårbarheter som kan drabba exempelvis ett samhälle eller en verksamhet. Bergreppet används bland annat av ovanstående myndigheter.

A.3 Bakgrund

Detta kapitel tar upp bakgrund som är relevant för rapporten.

A.3.1 Författarens bakgrund

Författaren till denna rapport har ingen formell utbildning inom samhällsfrågor eller etik.

A.4 Metod

Detta kapitel redovisar de metoder som användes för att besvara frågeställningarna som tidigare presenterats.

A.4.1 Litteraturstudie

Till denna rapport har en litteraturstudie genomförts. Detta för att få fram material att diskutera kring gällande ämnet samt frågeställningarna. För att genomföra litteraturstudien användes sökmotorn *Google Scholar*. Sedan lästes sammanfattningen för de fem första resultaten för att sålla bland materialet och avgöra om innehållet var relevant för studien. Sökord som användes var:

- OSINT
- OSINT google maps
- Aggregated data

För att hitta information om riskanalysmetoder besöktes svenska, statliga myndigheters hemsidor för att hitta dokument, rapporter och handböcker inom området. De myndigheter som besöktes var Myndigheten för samhällsskydd och beredskap, Totalförsvarets forskningsinstitut, Säkerheitspolisen och Försvarsmakten. Anledningen till att just dessa myndigheter valdes var för att dessa är stora myndigheter som riktat in sig på områden inom bland annat samhällsskydd och riskhantering.

A.4.2 Produktanalys

För att hitta information om en produkt som liknar Smart City valdes en av de populäraste produkten på marknaden, i detta fall Google Maps. Anledningen till att just Google Maps

valdes var för att den var lättillgänglig och finns för flera typer av plattformar, exempelvis mobil och dator. Produkten användes och funktionalitet dokumenterades.

A.5 Resultat

I detta kapitel presenteras de resultat som togs fram med hjälp av de studier som genomfördes.

A.5.1 Litteraturstudie

Detta kapitel tar upp de resultat från litteraturstudien som krävs för att kunna besvara frågeställningarna.

Metoder för att analysera samhällsrisker

FOI tog 2011 fram en handbok för att utföra en RSA. Handboken beskriver deras modell FORSA, en modell baserad på vetenskapligt belagda teorier, lagar och vägledning från MSB. Modellen ska kunna användas av aktörer inom krisberedskapsystem på samtliga nivåer för att kunna upprätta en RSA som följer rapportens krav och riktlinjer. Modellen är utvecklad för svenska kommuner, landsting och centrala myndigheter och baseras på fem stycken steg med tillhörande frågor:

- **Verksamhetsbeskrivning**
 - Vilka verksamheter bedriver organisationen?
 - Vilka är verksamheternas prioriterade åtaganden?
 - Vilka kritiska beroenden behövs för att upprätthålla de prioriterade åtagandena?
- **Riskbedömning**
 - Vilka oönskade händelser kan påverka verksamheten?
 - Vad är sannolikheten att de inträffar?
 - Vilka skulle konsekvenserna bli om de inträffar?
 - Vilka osäkerheter finns det i bedömningarna av sannolikhet och konsekvens?
- **Händelseanalys**
 - Hur sårbara är de kritiska beroendena?
 - Vilken förmåga har verksamheten att upprätthålla sina prioriterade åtaganden?
- **Åtgärder**
 - Vilka säkerhetsförbättrande åtgärder av särskilt intresse har redan genomförts?
 - Vilka ytterligare säkerhetsförbättrande åtgärder skulle kunna genomföras?
- **Arbetsredogörelse**
 - Hur genomfördes RSA?

Handboken redogör skillnaden mellan riskanalys och sårbarhetsanalys. En riskanalys lämpar sig bäst för händelser som går att värdera utifrån sannolikhet och konsekvens, en sårbarhetsanalys är bättre lämpat för sällsynta händelser där det är svårt att uppskatta både sannolikhet och konsekvens, möjligtvis för att en sådan händelse aldrig tidigare inträffat. Analyserna har även två olika syften, en riskanalys ska ses som ett beslutsunderlag för huruvida en risk ska kontrolleras, reduceras eller lämnas utan åtgärd medan en sårbarhetsanalys har som syfte att identifiera svagheter och analysera dess påverkan kopplat till en verksamhet.

Resultaten från en riskanalys ska enligt rapporten innehålla hur möjligt det är att upprepa, redovisa och kontrollera risken empiriskt. En sårbarhetsanalys bör besvara frågorna:

- Vad är skyddsvärt?
- Vad kan hota det skyddsvärda?
- Hur sårbart är det skyddsvärda?
- Hur är förmågan att stå emot och hantera påfrestningar på det skyddsvärda? [40]

MSB har även ett eget vägledningsdokument gällande RSA som också riktar sig mot kommuner, landsting, länsstyrelsen och centrala myndigheter. Detta dokument har en riskhantering som baseras på fyra huvudsteg med tillhörande understeg:

- **Utgångspunkter**
 - Roll och ansvarsområde
 - Metod, perspektiv och avgränsning
- **Riskbedömning**
 - Riskidentifiering
 - Riskanalys
 - Riskutvärdering
- **Sårbarhetsbedömning**
 - Förmågebedömning
 - Sårbarhetsanalys
- **Riskbehandling**
 - Resultat och slutsatser
 - Fortsatt arbete, åtgärder, planer

Enligt MSB finns även vissa grundläggande delar som bör finnas med i en slutgiltig RSA oavsett vilken metod som användes för att nå resultatet, dessa är:

- Att systematiskt kunna identifiera önskade händelser
- Kunna bedöma hur troligt det är att händelserna inträffar
- Kunna bedöma omedelbara, negativa konsekvenser från händelsen [41]

Risker med aggregerad data

FOIs handbok om FORSA beskriver riskerna som finns med att ta beslut utifrån aggregerad data, dessa risker är med avseende på en RSA. Det finns olika risker baserat på vilken detaljeringsgrad den aggregerade datan har. En låg detaljnivå ger en snabb överblick men kan lätt göra att väsentlig information blir svårare att identifiera. En hög detaljnivå kan leda till en övertrö på informationen, en välgenomförd och detaljerad analys kan dock göra det möjligt att ta informerade beslut. [40]

Försvarsmakten beskriver i sin handbok *Sekretessbedömning Del A* hur uppgifter, i detta fall hemliga, ska placeras för att skydda information. Handboken beskriver hur en bedömning av placering av information eller uppgifter ska göras. Bedömingen ska behandla ifall informationen förändras om den röjs tillsammans med annan information och ifall konsekvenserna blir större om informationen röjs ihop kontra enskilt. Handboken beskriver även att aggregering av uppgifter är något som ska uppmärksamas i en säkerhetsanalys av ett IT-system. [42]

Det finns tre fall av aggregation av uppgifter med avseende på placering i informationssäkerhetsklass:

1. Varken mängd eller kombination av uppgifter förändrar placering i informationssäkerhetsklass.

2. Genom att kombinera flera uppgifter erhålls nya uppgifter som inte hade varit möjlig att erhålla med tillgång endast till uppgifterna var och en för sig.
3. Mängden uppgifter förändrar placering i informationssäkerhetsklass.

Vilken klass informationsmängden faller inom avgör hur denna ska hanteras. Hos Försvarsmakten finns fyra olika säkerhetsklasser, klassen anger hur en uppgift eller handling ska hanteras för att erhålla rätt skyddsnivå och inte sekretessen för en uppgift eller handling. De olika klasserna är HEMLIG/TOP SECRET, HEMLIG/SECRET, HEMLIG/CONFIDENTIAL och HEMLIG/RESTRICTED, ju högre nivå desto starkare skydd krävs. [43]

Hot från antagonistiska grupper eller individer

MSB skriver i sitt vägledningsdokument rörande skydd mot antagonistiska hot och terrorism hur privata och offentliga aktörer ska stärka sitt skydd mot terrorattentat. Dokumentet tar upp hur en risk- och scenarioidentifiering och riskanalys ska genomföras, det går även igenom terrorattentats olika stadier:

- Målval
- Planering
- Övning
- Attack
- Exploatering

De stadier som är intressanta för denna rapport är främst planering och övning. Under planeringsfasen bedömer gärningspersonen eller -gruppen säkerhetsnivån på platsen där attentatet tänkts genomföras på. Detta sker genom informationsinhämtning, något som kan ske både på plats och via internet. Övningsstadiet har som syfte att dels bestämma vilka problem som kan uppstå under attentatet men även vilken respons attentatet förväntas få.

År 2021 kom Säpo fram till att terrorhotnivån i Sverige var på en skala 3 av 5, vilket betyder en förhöjd hotnivå. Denna terrorhotsbedömning är dock inte direkt överförbar på samtliga verksamheter utan är en bedömning av aktörers förmåga och avsikt att genomföra attentat på en nationell nivå. [44]

Open Source Intelligence

Open Source Intelligence, OSINT, innebär att sammanställa, analysera och dra slutsatser från öppen och tillgänglig information. Denna information kan komma från exempelvis bloggar, sociala medier eller publicerad myndighetsinformation. Ju större mängd information som hämtas desto större är sannolikheten att hitta relationer eller att kunna dra slutsatser baserat på detta. Ett problem är dock att det finns stora mängder information vilket gör att det är svårt att hämta denna manuellt. [45]

Det finns framtagna verktyg för att göra en så kallad OSINT-analys. Ett av dessa är *Maltego*. Verktyget hittar automatiskt öppen information om ett särskilt mål från bland annat sökmotorer, sociala medier och APIer. [46]

A.5.2 Produktanalys

I detta kapitel presenteras resultat från produktanalysen som genomfördes.

Tillgänglig information hos Google Maps

När rapporten skrevs tillhandahöll Google Maps användaren information om var exempelvis restauranger, museum och butiker finns placerade, det går även att se öppettider för dessa. Det finns även möjlighet att på många av dessa platser se hur många besökare som förväntas

vid olika tider på dygnet samt hur många besökare det faktiskt är. Det går inte att se exakt antal besökare, utan bara en uppskattning som är relativt hur många besökare en plats brukar ha. Det är även möjligt att få vägbeskrivningar till samtliga platser på kartan, här går det att välja transportmedel samt se ifall det är köer. [47]

A.6 Diskussion

I detta kapitel diskuteras resultatet och metoder som används för att få fram resultatet.

A.6.1 Metod

För att hitta information om RSA-metoder besöktes svenska myndigheters hemsidor, detta ledde till att metoder som används av länder utanför Sverige, främst länder utanför Europa, inte beaktats i denna rapport. Ett annat problem med att använda svenska myndigheter som källor är att en stor del av informationen som myndigheterna själva arbetar efter är opubllicerad eller sekretessbelagda, detta gör att vissa moment av metoderna omedvetet kan ha missats.

Ett problem med att genomföra en produktanalys är att det är stor sannolikhet att missa funktionalitet hos produkten, särskilt en produkt med många funktioner så som Google Maps har.

A.6.2 Metoder för att genomföra en risk- och sårbarhetsanalys

Metoderna som hittades för att genomföra en generell RSA innehöll i princip samma steg och delar:

- Att utföra en verksamhetsanalys, vilka delar av verksamheten är värda att skydda? Finns det kritiska beroenden inom verksamheten?
- En riskbedömning. Vilka öönskade händelser kan drabba verksamheten och hur stor är sannolikheten att detta inträffar? Vad skulle konsekvenserna av att de inträffar bli?
- En sårbarhetsbedömning. Hur sårbara är de kritiska beroendena som finns inom verksamheten?
- Ta fram åtgärder, vilka åtgärder ska genomföras?

Samtliga metoder beskrev hur aktörer och verksamheter bör genomföra en RSA, dessa metoder är även applicerbara på annat än verksamheter då det är möjligt att göra samma typ av analys på exempelvis en hel stad eller ett land. Samtliga RSA-metoder går ut på att identifiera vad som är vär att skydda, på vilket sätt det skyddsvärda kan utsättas för skada samt vad konsekvenserna av en sådan skada blir.

Under studien hittades inga riktlinjer gällande hur specifikt aggregerad data och dess risker kopplade till samhället ska analyseras. Det är dock fullt möjligt att kombinera de riktlinjer Försvarsmaktens handbok *Sekretessbedömning Del A* om hur aggregerad data ska hanteras för att undvika att erhålla ny information tillsammans med en RSA-metod för att analysera samhällsrisker. Det är möjligt att kombinera dessa metoder då dem har använts av myndigheter i praktiken.

Det skulle alltså vara möjligt att analysera vad riskerna med den aggregerade datan är och därefter analysera hur dessa risker påverkar olika delar av samhället.

A.6.3 Oavsiktlig användning av Smart City ihop med andra produkter

Det finns en handfull funktioner som skiljer Smart City och Google Maps, en av dessa är att kunna se exakt antal personer som passerat en plats, denna funktionalitet finns i Smart

City men inte i Google Maps. Då MSB beskriver att informationshämtning är en viktig del av planeringsfasen kan det vara sannolikt att individer eller grupper som har avsikt att begå exempelvis terrorattentat har intresse i att ta del av information som innehåller hur många männskor som brukar befina sig på vissa platser under vissa tider av dygnet. Det kan även göra det lättare att bilda sig en uppfattning av hur samhället reagerar på vissa händelser då exempelvis alla rapporterade händelser hos Polisen samlas in, evenemang som sker i stan samt antal männskor på vissa platser.

Även om Smart City bara använder sig av data som redan ligger ute för allmänheten är det viktigt att komma ihåg att ihopsamlad data kan skapa ny information så som Försvarsmakten skriver i sin handbok *Sekretessbedömning Del A*. Det är alltså viktigt att analysera hur datan som finns i Smart City förändras, både tillsammans med annan data i produkten men även data som finns hos andra produkter. Att enbart fokusera på att den data som Smart City använder sig av redan finns tillgänglig för allmänheten kan därför vara ett felaktigt argument för att hävda att Smart City inte tillför risker. Det är inte hållbart att endast fokusera på vad datamängder kan tillhandahålla för information på egen hand, precis som Försvarsmakten skriver i sin handbok är aggregering av uppgifter något som ska finnas med i en säkerhetsanalys av ett IT-system.

Skulle det även implementeras i produkten att det blir lättare för användaren att själv lägga till information eller källor de är intresserade av kan även användare utan tekniska kunskaper använda systemet för sin egna fördel, något som utökar produktens användningsområden för en stor grupp. Med tanke på att Sverige har en förhöjd terrorhotsnivå finns det alltså intresse hos individer eller grupper att planera exempelvis terrorattentat i landet. Att samla information som antagligen ligger i deras intresse skulle göra det lättare att planera upp ett terrorattentat. Med ett OSINT-verktyg kan användaren hitta nya, kanske inte helt uppenbara, datakällor för att utöka informationsmängden ytterligare. Används exempelvis verktyget Maltego som hittar öppna APIer är dessa sedan möjliga att implementera i Smart City.

Det går därför att argumentera för att Smart City förenklar planeringsfasen för exempelvis terrorgrupper i Sverige. Att göra det lättare att se var folksamlingar brukar ske och om dessa folksamlingar korrelerar med andra händelser är gissningvis av intresse för grupper med mål att skada samhället.

A.7 Slutsatser

Här presenteras svar på rapportens frågeställningar baserat på resultat och diskussion.

Vilka metoder finns det för att analysera samhällsrisker gällande aggregerad data?

Det finns framtagna och etablerade metoder för att utföra risk- och sårbarhetsanalyser för kommun, landsting eller statlig myndighet eller andra verksamheter, bland annat FOIs modell FORSA och MSBs vägledning. Det finns även metoder för att analysera hur information förändras tillsammans med annan information och vad det kan ha för risker. Denna rapport hittade dock inte vedertagen metod för att analysera samhällsrisker med specifikt aggregerad data, det är dock möjligt att kombinera metoderna i de olika rapporterna för att uppnå detta.

Finns det scenariet där Smart City, tillsammans med andra produkter som samlar liknande information, ökar risken för oavsettlig användning av antagonistiska grupper?

Det går att argumentera för att Smart City kan förenkla planeringsfasen av ett terrorattentat, något som kan vara av intresse hos antagonistiska grupper. Detta då Smart City kan tillhandahålla annan information än det som redan finns på exempelvis Google Maps.



B

Unreal Engine: Mer än bara underhållning - Filip Josefsson

B.1 Introduktion

Smart City använder Unreal Engine 4 för att visa en karta och sedan placera markörer på den. Detta är dock långt ifrån vad man föreställer sig en spelmotor är gjord för, men spel är inte det enda motorn används till. Vad denna rapport vill undersöka är vad annat som Unreal Engine används till förutom spelutveckling, filmproducering och annan underhållning.

Den första versionen av Unreal Engine (förkortat UE) släpptes år 1998 tillsammans med spelet *Unreal* för att visa vad spelmotorn kunde åstadkomma [48]. Sedan dess har UE uppdaterats och är just nu på dess femte version [49]. UE byggdes för att skapa spel men har använts till mycket mer. I dagens värld finns företag som utnyttjar vad som uppfanns för spel och applicera det i nya användningsområden i vad som kallas "gamification". Detta skulle kunna hänga ihop med hur spelmotorer används i nya syften. Epic Games gör själva reklam om dessa användningsområden när de visar hur motorn används till att, bland annat, skapa filmer och visuallisera arkitektur [50]. Därför gjordes en fallstudie med målet att upptäcka fler intressanta användningsområden som motorn används till.

B.1.1 Syfte

Syftet med den här rapporten är att undersöka vad Unreal Engine används till praktiskt utanför underhållning. Vad som därmed inte tas upp är bland annat spelutveckling och filmproducering. Utöver det ska även varför den används där hittas, tillsammans med de konsekvenser som uppstår av det valet.

B.1.2 Frågeställningar

1. Vad har Unreal Engine använts till förutom underhållning?
2. Vilka anledningar ledde dessa projekt till att välja Unreal Engine och vilka konsekvenser hade det valet?

B.2 Teori

Detta kapitel tar upp den relevanta teori kring motorn Unreal Engine som är användbar till att förstå denna rapport.

B.2.1 Unreal Engine

Unreal Engine är en spelmotor från Epic Games. Den första versionen av motorn släpptes år 1998 tillsammans med datorspelet *Unreal* [48]. Sedan dess har den uppdaterats och i april 2022 släpptes motorns femte iteration, Unreal Engine 5 [49]. I denna individuella rapport behandlas flera versioner av motorn. Av den anledningen skrivs en siffra med i förkortningen när vilken version av motorn vill framhävas, till exempel UE4.

Licensiering

Hur de olika undersökningarna och projekten som beskrivs i denna rapport fått tag på själva motorn varierar. I vanliga fall kostar det att använda andras program men Epic Games har länge tillåtit flera sätt att använda motorn gratis. Vid köp av spelen *Unreal Tournament 2004* till och med *Unreal Tournament 3* ingår en editor av respektive spels motor som tillåter folk att gratis göra modifikationer av spelet eller icke-kommersiella produkter [51, 52]. Det finns även en demo-version av UE2 [53]. Under tiden när UE3 var den senaste släpptes även ett gratis editor av motorn som heter *Unreal Development Kit* (UDK) [54]. Från och med UE4 är det däremot möjligt att ladda ner spelmotorn och använda den gratis till icke-kommersiella syften [55].

Programmeringsspråk

Från UE1 till och med UE3 var motorns programmeringsspråk *Unrealscript* [56]. Det var ett objektorienterat språk som liknande C++ och Java och var gjort för programmering i spelmotorn. Motorn bytte språket däremot när UE4 släpptes. UE4 och senare versioner programmeeras i C++ eller i *Blueprints*, ett visuellt programmeringsspråk gjort för motorn [57].

B.3 Metod

För att kunna besvara denna raports frågeställningar utfördes en fallstudie. Studien gjordes genom att söka efter relevanta rapporter på söktjänsten Google Scholar. På söktjänsten söktes rapporter från projekt som utnyttjat UE utanför underhållningssyften, som till exempel utbildning. Därefter gjordes ett urval.

Det första urvalet bestod framförallt av att välja bort de rapporter som var svåra att komma åt, till exempel rapporter krävde ett köp av en fysisk bok. Anledningen till detta var dels att projektet inte hade en budget och dels att denna rapport hade strikta tidskrav. Vilken version av spelmotorn som användes till projekten i rapporterna spelade ingen roll. Därefter uteslöts rapport med användningsområden som ansågs vara för lika andra, redan upptäckta, användningsområden. Syftet med detta var att öka antalet distinkta användningsområden som kunde tas upp i denna rapport. Slutligen, om en rapport utnyttjade UE men visade sig varken diskutera eller kommentera motorn, på ett sätt som kunde besvara rapportens frågeställningar, uteslöts även den rapporten.

Vid varje sökning genomsöktes de första 5 sidorna på Google Scholar. På varje sida lästes först rapporternas rubriker och sedan deras sammanfattningar ifall de såg ut att kunna passera urvalet. Om även sammanfattningen var relevant till denna rapport lästes rapporten mer genomgående. Sökorden som användes var:

- Unreal engine applications

- Unreal engine non video game uses
- Unreal engine serious game

B.4 Resultat

I detta kapitel beskrivs de rapporter som har upptäckts i fallstudien. Beskrivningen består dels av vad för sorts projekt som rapporterna handlar om och dels den relevanta informationen som kan besvara denna rapports frågeställningar. Sist i kapitlet finns en sammanfattning av det som ansågs vara viktigast för att besvara frågeställningarna.

B.4.1 Synnedsättningar

I projektet *Simulating visual impairments using the Unreal Engine 3 game engine* har UE3 använts för att simulera synnedsättningar [58]. Programmet som beskrivs i rapporten visade först ett realistisk café. Därefter utvecklades filter som skulle simulera nedsättningarna. I det slutliga programmet kunde de simulera sex stycken synnedsättningar. Målet med programmet var att skicka det till skolor eller andra organisationer för att utbilda folk om synnedsättningar.

Anledningarna till att de valde UE var framförallt hur motorn var bra på realistisk rendering, gick snabbt att använda och var billig. Programmet ansågs även vara enkelt att distribuera till organisationerna som skulle vilja använda programmet.

En av konsekvenserna som rapporten tar upp var att inlärningskurvan för motorn var brant, däremot fick de ändå klart en prototyp på enbart några veckor. Utöver inlärningskurvan poängterades även färgändring och spelet som modifierats som problem med motorn. Motorn hade inget sätt att ändra färger i efterhand var ett problem eftersom de därmed inte enkelt kunde simulera färgblindhet. En unik konsekvens uppstod av att de fick tag på utvecklingsprogrammen genom att köpa och modifiera spelet *Unreal Tournament 3*. Eftersom programmet framförallt var en modifikation av det spelet, som är ett skjutspel med en 18-årsgräns, skulle det vara opassande att ladda ner spelet och modifikationen till skolor. Detta begränsade vilka de skulle kunna utbilda med hjälp av detta programmet.

B.4.2 AILiveSim

AILiveSim är en simuleringsplattform byggd för AI och autonoma system. Programmet har målet att tillåta testning av prototyper och koncept, träna AI, optimering av algoritmer och att validera produkter [59]. I rapporten står inte vilken version av UE som används men baserat på att rapporten publicerades 2019 och att UE4 släpptes 2014 är det mest troligen UE4. En av sakerna som simuleras i programmet är sensorer. Detta är för att ett av de mer specifika användningsområdena som tas upp i rapporten är träning av självkörande bilar.

I rapporten skrevs nästan bara om de rent tekniska aspekterna av UE som anledning till valet av spelmotor. Dessa anledningarna var bland annat de lovade prestanda uppdateringarna, en god bildgenerator och renderingspipeline, kvaliteten på andra användares open source-tillägg och möjligheten till att använda klient-server-arkitektur. Ingenstans i rapporten nämns grafiken av självaste spelmotorn.

En konsekvens av valet av motor var däremot att motorn prioriterade bildfrekvens över fysiksimuleringar. Detta leder till att de inte helt kunde lita på simuleringen när den ersätter trogen fysiksimulering mot prestanda.

B.4.3 Arkitektutbildning

I undersökningen *Videogame Technology in Architecture Education* användes UE4 för att visa arkitekters byggnader [60]. I rapporten undersöks framförallt hur folk kände kring användning

av spelmotorer vid arkitektutbildning. För detta gjordes en virtuell yta med en byggnad som användare av programmet kunde gå runt och in i från en människas naturliga perspektiv. Anledningen till att rapporten tar upp detta är, förklarar de, att andra arkitektprogram visar byggnader från ett fågelperspektiv. Deras slutsats var att folk uppskattade programmet och att i framtiden skulle de göra programmet mer interaktivt.

Rapporten beskriver vissa av anledningarna till att UE valdes. Fördelarna med UE var att motorn använde 3D-modellering och annan teknik som liknar det som redan används inom arkitektur, är gratis och hade programmeringsspråk som var vänligare till ickeprogrammära. Författarna avslutar rapporten med en jämförelse mot spelmotorn Unity där de kallar UE mer konstnärvänlig. Rapporten tar dock inte upp några nämnvärda konsekvenser med valet av UE.

B.4.4 Interoperabilitet

Simulation Interoperability with a Commercial Game Engine är en undersökning från 2005 som undersöker interoperabiliteten av simulering i spelmotorer [51]. Rapporten tar upp hur spelmotorer hade gjort stora framsteg inom visualisering, användargränssnitt och interaktion i förhållande till klassiska simuleringsprogram. Därför ville de undersöka hur väl spelmotorn UE2 var på att simulera och hur likt simuleringen fungerar i jämfört mot andra simuleringsprogram, vilket är interoperabilitet. I deras experiment simulerades fysiologin av folk som drabbas av att vara med i en flygplanskrasch.

Anledningarna till att de valde UE var att den hade realistisk grafik, ett bra användargränsnitt, möjlighet till fysisk simulering och var billig.

I och med att rapporten framförallt var en undersökning av hur väl spelmotorn fungerar som simuleringsverktyg hade de en del kritik kring konsekvenserna av valet av motor. Ännu en gång bör det sägas att denna kritik avser en 17 år gammal version av UE2. Först och främst var att simuleringarna inte hade någon riktig interoperabilitet med andra simuleringsmjukvaror. Detta innebär att simuleringarna inte skedde på samma sätt utanför UE, vilket gjorde den svårare att använda i samspel med andra simuleringsprogram. Den andra kritiken var att programmeringsspråket UnrealScript inte var passande till simulering.

B.4.5 Kärnkraftverk

I rapporten *Unreal III based 3-D virtual models for training at Nuclear Power Plants* utforskades UE3 för att hitta funktioner som skulle kunna vara användbart vid olika simulerningar av kärnkraftverk [61]. Specifikt träning och utbildning tas upp som användningsområden. Rapporten blandar funktioner som ingick i självaste motorn och implementationer som de själva hade utvecklat när de rapporterar sina upptäckter. Detta ledde till att det blev svårare att avgöra vilka funktioner som tillhörde UE och vilka som utvecklades till syftet av rapporten.

Rapporten listar 12 aspekter i implementation av deras program som var användbara till kärnkraftverkssyften. Många av de liknar varandra vilket ger en förkortad lista på:

- Den tillåter flera användare vara i samma karta.
- Den har goda visuella och interagerbara funktioner.
- Den har ickespelarkaraktärer (NPCs).

Den tar dock inte upp någon nackdel med användning av UE till deras syften.

B.4.6 Evakueringssimulering

I undersökningen *Using a Game Engine for VR Simulations in Evacuation Planning* från 2008 utforskas användning av spelmotorer för simulering av evakueringssplaner [53]. I deras undersökning låt de testdeltagare evakuera en byggnad i UE2 och mätte hur lång tid det skulle ta att evakuera den byggnaden jämfört mot en simulering i en riktig byggnad. Det kom fram till att de tog nästan lika lång tid i båda fallen, vilket innebar att den virtuella simuleringen skulle kunna vara ett bra sätt att testa evakueringssplaner.

Undersökarna hade några skäl till att välja UE. Motorn var gratis för akademisk användning, hade tillgång till flera spelare på samma karta och hade en robust 3D-rendering.

Ett problem som sedan uppstod var att vissa testpersoner inte förstod kontrollerna på direkten. Ett annat problem var att, i deras implementering, krockade användare med varandra när de försökte gå igenom samma dörr samtidigt och kunde därmed inte ta sig igenom.

B.4.7 Allmän utbildning

Using Virtual Reality in Education är en rapport som går igenom hur Virtual Reality (VR) kan användas i utbildning. Programmet som utvecklades var gjort på UE4 för att skapa VR-miljöerna. Till denna rapport skapades fyra program med tanken att de skulle visa olika utbildningsområden [62]. Programmen som skapades var:

- Ett program som visar information om varje ben i skelettet.
- Ett program som visar vårt solsystems planeter i skala.
- En interaktiv byggnad för arkitektur.
- En visualisering av kärnkraftverket i Dukovany.

Fördelarna de tar upp med att välja motorn var möjligheten att programmera i C++ och att UE har realistisk grafik. I största allmänhet anses ändå rapporten vara relativt bristande i förhållande till de andra. Bland annat framhäver författarna att VR kan utnyttjas inom alla vetenskapliga discipliner trots att de endast har visat fyra begränsade områden. Den tar inte heller upp några konsekvenserna av motorvalet.

B.4.8 Ljussimulering

I rapporten *A calibration methodology for light sources aimed at using immersive virtual reality game engine as a tool for lighting design in buildings* undersöktes hur väl UE4:s ljussimulering är för ljusdesign [63]. Vad de kom fram till var att motorn simulerar ljus väl, men inte lika bra som program gjorda för ljusdesign.

Artikeln gör ingen särskild förklaring till varför de valde UE. Snarare togs UE som ett prov på hur spelmotorer fungerar i ljusdesign. En konsekvens av valet till motor var först att dokumenteringen av motorn inte var tillräckligt utförlig för deras syften. Ett annat problem de fick var att varje ljuskälla har cirka 30 parametrar. Detta anser de är troligen användbart för spelprogrammering men överflödigt vid ljussimulering.

B.4.9 Sammanfattning av fallstudie

UE har använts till många områden men framförallt går det att dela upp rapporterna i två kategorier av syften: Utbildning och simulering. De texter som använder motorn för utbildning var rapporterna om synnedsättningar, arkitektutbildning, kärnkraftverk och allmän utbildning. Resterande texter kan ses som simuleringar.

Utbildning

Flera av rapporterna som hittades i fallstudien beskrev projekt som skapade program i utbildningssyfte, men vad som skulle utbildas skiljer sig. Allt från att simulera synnedsättningar till hur kärnkraftverk fungerar. Anledningarna till att dessa projekt valde UE var framförallt att den var billig, relativt enkel att använda och hade realistisk grafik.

Bara en av rapporterna om utbildningsprogram tar upp vad konsekvenserna av att välja motorn var. Denna var synnedsättningsrapporten som troligen gjorde programmet som mest liknade en simulering bland utbildningsprogrammen. Konsekvenserna för de var att UE3 inte hade enkla funktioner för att simulera färgblindhet och att de behövde få tag på motorn genom ett spel med 18-årsgräns, som är opassande för deras syften.

Simulerings

Rapporter som handlade om simuleringar hade en stor bredd i vad som simulerades. Från att simulera sensorer på självkörande bilar till hur väl en evakueringsplan fungerar. Även bland dessa rapporter nämndes att UE var gratis och att den hade realistisk grafik som anledningar till att välja motorn. Utöver de anledningarna nämndes även motorns fysiksimuleringar, open source-tillägg och prestande uppdateringar.

Rapporter som handlade om simuleringssystem som byggts med UE tog ofta upp, till skillnad från utbildningsrapporterna, konsekvenser och problem med motorvalet. En konsekvens av att välja UE var enligt en rapport att motorn offrade noggrannheten av dess fysiksimulering för att behålla god prestanda, vilket sänker trovärdigheten på simuleringens exakthet. En annan konsekvens är att vissa inte förstod hur motorn fungerar tillräckligt på grund av att dokumenteringen inte var tillräckligt. Något som sticker ut efter att ha gått igenom dessa rapporter är att motorn fungerar väl fram tills exakthet och noggrannhet krävs eller när folk försöker tillämpa den i specialiserade områden, där dedikerade program fungerar bättre.

B.5 Diskussion

Här nedan diskuteras metoden som valdes för denna studie och dess resultat.

B.5.1 Upptäckter under fallstudien

Under fallstudien hittades uttrycket "Serious gaming" som handlar om att utnyttja kunskapserna och teknikerna som finns i spelindustrin och överföra dem till praktiska användningsområden. Detta är dock en tolkning eftersom texterna inte hade en och samma definition. Efter att uttrycket upptäcktes användes det som ett sökord i fallstudien. Dock gav den inte lika bra resultat som förväntat. Detta beror troligen på att uttrycket är inte särskilt vidspärrat och att ordet "game" ingick i sökningen, vilket ledde till att många spelartiklar som inte är "Serious gaming" dök upp istället.

I metoden skrevs det att rapporter inom lika användningsområden skulle väljas bort. Detta visade sig vara ett onödigt urval eftersom rapporterna som hittades visade sig vara mer olika än förväntat. Förväntning var att de enda texterna som skulle hittas var vissa fysiksimuleringar och utbildning kring VR. Därför sattes det urvalet in i metoden, för att undvika massa texter om dem ämnena. Det var förvånande hur specifika program och tillämpningar som fanns i rapporterna som hittades. Framförallt rapporten om synnedsättningsprogrammet visade hur specifika och intressanta tillämpningar folk har hittat i vad som bar är en spelmotor.

B.5.2 Problem från metodvalet

Ett problem som dock uppstod från metoden i denna rapport var valet att ta med flera generationer av motorn. I fallstudien hittades 4 rapporter som använde UE4, 2 som använde

UE3 och 2 till som använde UE2. Problemet uppstod när några av de äldre rapporterna tog upp problem som antingen kan ha eller faktiskt har förbättrats. Till exempel fick UnrealScript kritik men eftersom den inte längre används är den kritiken inte relevant för dagens version av motorn. Om en rapport i framtiden vill fokusera på konsekvenserna och problemen med att använda dagens version av UE måste gamla rapporter uteslutas.

Ett annat problem var att under studien hittades flera texter som ansågs vara intressanta och relevanta till delar av frågeställningarna, men som inte hade en genomgående förklaring till valet av UE. I många rapporter var förklaringen till valet väldigt kort, bara några få meningar. Därefter var det många som slutade nämna ens namnet på motorn efter de första kapitlen. Detta beror troligen på att flera av studierna brydde sig mer om vad de implementerat än vad motorn hade att erbjuda. Resultatet av det var att hälften av rapporterna inte gav någon intressant information om nackdelarna kring UE.

Vad som kunde ha hjälpt till att hitta fler konsekvenser och problem av motorvalet skulle kunnat vara att även ta med någon eller några rapporter som valde bort UE från deras studie. Även om de rapporterna inte skulle kunna besvara vissa av frågeställningarna skulle de troligen kunna ge en bättre diskussion kring åtminstone en av dem. Däremot ansågs att de texter och deras information som har hittats genom studien var tillräckliga för att ge goda svar till frågeställningarna.

B.5.3 Anledningarna till att välja motorn

Nästan alla rapporter som listar anledningen till valet av motor tar fram att den antingen är billig, numera gratis, har realistisk grafik eller både och. Dessa tycks vara de två viktigaste anledningarna till att välja UE. Till all användning som inte är kommersiella datorspel, TV-spel eller mobilspel är det gratis att använda UE. Till och med när motorn inte var gratis kostade det bara priset av ett datorspel. Detta kan ha lett till en ökad användning av motorn till skillnad från om det hade kostat. UE framstår som, baserat på dessa texter, en riktigt bra spelmotor på realistisk grafik. Det kan förvänts att de ser det som viktigt att ha realistisk utseende på programmen när de ville antingen utbilda eller simulera verkliga situationer. För att göra länken mellan vad som händer på skärmen och vad som skulle hända i verkligheten så tydlig som möjligtvore ett realistisk utseende antagligen att föredra.

Bland de mindre vanliga anledningarna till valet av motor var, bland annat; flerspelarfunktionalitet, bra programmeringsalternativ och att andra användare har lagt till flera funktioner till motorn genom open source. UE:s flerspelarfunktionalitet framstår som en fördel som finns från att UE är en spelmotor, eftersom många spel har flerspelarlägen. Vad gäller dess programmeringsalternativ finns det dock intressanta aspekter kring detta. För det första var detta framförallt något som skrevs om UE4 till skillnad från äldre versioner. För det andra uppskattades båda programmeringsalternativ: C++ och Blueprints. C++ tycks vara mer uppskattat bland mer tekniskt inriktade personer medan resten, bland annat arkitekterna, föredrog den visuella programmeringen som Blueprints gav. Genom att erbjuda båda alternativ tycks Epic Games tillåta både folk som har och inte har programmeringsvana möjligheten att använda motorn. Sist var det open source-funktionaliteten som var uppskattad. Att ha möjligheten till att kunna utöka motorns möjligheter, tillsammans med alla guider många tidigare användare har gjort, var uppskattat.

I bilaga C: *Spelmotorers påverkan på designprocessen - Daniel Kouznetsov*, har en annan rapport skrivits som tar upp andra praktiska områden som inte tas upp i denna rapport. För den intresserade kan den bilagan bidra till diskussionen om vilka praktiska användningsområden UE har använts i.

B.5.4 Konsekvenser av motorvalet

Den mest genomgående problemet folk hade med UE i deras projekt var att motorn inte hade specialiserade funktioner. Det finns en tydlig gräns mellan vad motorn är bra på och vad den inte var gjord för. Utvecklarna av simulatorn av synnedsättningar hittade inget inbyggt sätt att simulera färgblindhet, vilket troligen behöver speciell funktionalitet som Epic Games troligen inte hade förväntat sig. Forskarna som undersökte motorns duglighet som ljusdesignssimulator kom fram till att även om ljussimuleringen är bra finns bättre verktyg som är designade i syftet av ljussimuleringar. Som en spelmotor är troligen inte det viktigaste att ljuset ska vara så verklighetstroget som möjligt så länge det är realistisk nog för spelare. Utvecklarna av AILiveSim gillade inte att motorn hellre offrar fysisk noggrannhet för prestanda. Detta är i många fall motsatsen till vad ett spel vill ha vilket skapar ett problem med att använda en spelmotor på det viset. När UE används som en simulator tycks det som att den brister i för specialiserade områden.

B.5.5 Problem med resultatet

Ett problem med konsekvenserna som nämndes i vissa rapporter var att vissa hade inte särskilt relevanta konsekvenser. Till exempel i rapporten om simuleringen av evakueringsplanerna nämndes kontrollerna och krockdetekteringen som problem. De två problemen är troligen enbart implementationsfel som egentligen inte hade med motorn att göra. Däremot gjordes beslutet att ta med de i denna rapport för att även om de berodde implementationsfel skulle dessa fel möjligen inte skett om de valt en annan spelmotor. Ovanpå implementationsproblemen fanns även de problemen som numera är utdaterade. I interoperabilitetsrapporten kritiseras UnrealScript men eftersom det språket inte längre används, utöver att det finns ingen riktig anledning till att välja äldre versioner av spelmotorn över de nya, är hur relevant denna information diskuterbart. Den informationen skrevs i denna rapport ändå eftersom metoden inte satte en begränsning och tidigare konsekvenser av motorvalet kan ses som intressanta.

B.6 Slutsatser

Här besvaras frågeställningarna som sattes i början av den här individuella rapporten.

B.6.1 Vad har UE använts till förutom underhållning?

Utifrån de rapporter som hittades under fallstudien går det att dela användning av Unreal Engine i två kategorier: utbildning och simulerings. Utbildningen kan bestå av saker från att visa hur folk med synnedsättning ser världen, till hur arkitekters byggnader och mänskliga skelettet ser ut. Simulerings var minst lika vanliga. I denna raports studie hittades simulerings av evakueringar, kärnkraftsverksarbete, ljusdesign och även simulerings gjorda för att träna AI med mera.

B.6.2 Vilka anledningar ledde dessa projekt till att välja Unreal Engine och vilka konsekvenser hade det valet?

Först och främst valdes Unreal Engine för motorns realistiska grafik och att den var gratis. Den realistiska grafiken tycks vara uppskattad inom nästan alla områden. Andra skäl spelmotorn valdes var också att man kunde ha flera användare i samma område samtidigt, bra alternativ till programmeringsspråk och att den var snabb att använda efter man har kommit förbi inlärningskurvan.

Några konsekvenser av valet var bland annat en brist på funktionalitet i specialiserade områden. Bland annat kunde de som simulerade AI inte fullt lita på den fysiska korrektheten av

programmet eftersom spelmotorn prioriterar prestandan över fysiksimuleringen. Utvecklarna av ett synnedsättningsprogram saknade ett sätt att simulera färgblindhet. De som undersökte UE:s ljussimulering kom fram till att hur ljus fungerar i spelmotorn inte är lika exakt som de programmen en ljudsdesigner redan använder.



C Spelmotorers påverkan på designprocessen - Daniel Kouznetsov

C.1 Introduktion

I projektet har spelmotorn Unreal Engine använts för att visualisera händelser i en stad. Den här delen av rapporten ämnar sig att svara på hur man kan utnyttja spelmotorer för att få en mer effektiv designprocess och hur Virtual Reality (VR) kan komma till användning för att skapa en mer interaktiv upplevelse i processen.

C.1.1 Syfte

Syftet med denna rapport är att svara på frågan hur moderna spelmotorer, mer specifikt Unreal Engine, och VR används i designprocesser samt i vilka branscher detta förekommer. Detta kommer göras genom att studera hur Unreal Engine och liknande verktyg används idag för att ta fram prototyper.

C.1.2 Frågeställning

1. På vilka sätt kan spelmotorer förändra designprocessen?
2. Hur kan VR användas för att skapa en mer interaktiv designprocess?
3. Vilka branscher använder sig av spelmotorer som Unreal Engine i sina designprocesser?

C.2 Teori

Detta avsnitt beskriver begrepp och termer för att läsaren ska förstå resterande delar av rapporten.

C.2.1 Virtual Reality (VR)

VR låter en användare interagera med en fiktiv tredimensionell värld genom headset, glasögon och handhållna interaktiva enheter. VR brukar främst användas för spelande där användaren får en uppslukande och nästintill verklig upplevelse.

C.2.2 Spelmotor

En spelmotor är mjukvara designad för att låta utvecklaren skapa spel till olika plattformer som PC, konsol och telefoner. Vanligtvis innehåller en spelmotor bibliotek och andra tillägg som möjliggör och förenklar utvecklandet. [64]

C.2.3 Haptisk feedback

När det kommer till teknik är audiovisuell feedback vanligt, du har en skärm att titta på och det kommer även ljud. Haptisk feedback lägger till ett sinne, känsel. Principen för haptisk feedback är att det lägger till känsel i form av vibrationer och tryck. Vanligtvis används haptiska gränssnitt för uppgifter som utförs med händerna [65]. Ett haptiskt gränssnitt är exempelvis en handske som känner av kollisioner från någon extern enhet som en dator och skickar vibrationer och tryck till användaren.

C.2.4 Datorstödd konstruktion (CAD)

Datorstödd konstruktion (CAD) innebär att användaren utnyttjar en dator i samband med någon mjukvara gjord för ändamålet för att skapa och modifiera en design. [66]

C.3 Metod

I detta avsnitt beskrivs de metoder som används för att besvara frågeställningarna.

C.3.1 Metodval

Den metod som valts för att besvara frågeställningarna är en litteraturstudie men också egna erfarenheter från Unreal Engine. En litteraturstudie innebär att kvalitetsgranskad och konkret fakta om ämnet används för att kunna svara på frågeställningarna. Då spelmotorn har använts i projektet kan även skribentens personliga erfarenhet av den vara till hjälp.

C.3.2 Litteraturstudie

Den sökmotor som användes var universitetsbibliotekets egna på Linköpings universitet [67] som gav tillgång till hundratals databaser såsom Web of Science, Scopus, ScienceDirect och PubMed. Källor som används har funnits genom ett antal söktermer, dessa är:

- Unreal Engine
- Unreal Engine prototyping
- Unreal Engine vehicle
- Digital prototyping

Urvalet av källor gick till genom att gå genom sökresultaten och snabbt filtrera genom en överblick av rubrikerna för att se vilka som passar till ämnet. När ett antal till synes relevanta källor valts utifrån rubriker lästes sammanfattningar för att ytterligare smala ner källorna. Till en början togs 25 källor fram men när rapporternas relevans började avta för att besvara frågeställningarna togs sedan fem stycken källor fram från det första urvalet. Dessa fem valdes genom att tänka över relevansen av rapporterna med hänsyn till frågeställningarna. Varför fem rapporter valdes är för att de källor som valdes var tillräckligt för att besvara frågeställningarna.

C.4 Resultat

I den här delen av rapporten framställs den information som tillkommit från litteraturstudien.

C.4.1 Spelmotorer, nya tidens CAD-verktyg?

Sen CAD börjat användas har spelmotorer varit med och drivit industrin för 3D design framåt [68]. En mycket viktig del när det kommer till design är korrekthet, det som ska visas på skärmen ska stämma överens med verkligheten. Mjukvara som används för design kräver en renderingsprocess som kan ta tid. I spelmotorer renderas det som visas på skärmen i realtid. För att få en realistisk rendering används matematiska genvägar som gör resultatet verklighetstroget [69]. Resultatet blir en snabb rendering som designers kan ha nytta av, exempelvis sparande av tid. Det har förstås sina nackdelar, en mindre exakt rendering jämfört med renderingsmetoder som använder sig av andra mer tidskrävande metoder. Med utvecklingen som gjorts är det nu möjligt att få snarlika resultat för realtidsrendering och därför kan spelmotorer nu användas för att skapa realistiska konstruktioner inom exempelvis arkitektur.

Trots att CAD har förenklat designprocessen för designers visar studier på att det inte bara är positivt. En studie som analyserade litteratur på design i olika CAD-miljöer visade på att användandet av mjukvaran kan bidra till mindre kreativitet och brist på motivation. En lösning på problemet som framfördes är att använda sig av funktioner som går att skapa i spelmotorer, exempelvis att skapa mer intuitiva och engagerande sätt att skapa sin design på. Ytterligare argument för att försöka skapa mer i spelmotorer är att de flesta har inbyggda funktioner för samarbete, flerspelarfunktioner, vilket möjliggör för effektivt arbetande tillsammans med kollegor. [70]

C.4.2 Arkitekters utmaning

Att sälja in ett projekt till en kund krävs att kunden blir övertygad om att det som köps är av värde och att det är det bästa alternativet om det skulle finnas flera. Därför är en av arkitekturens största utmaningar att få till en slagkraftig framläggning för sin konstruktion. Ett byggnadsköp är en oerhört stor investering därför är det viktigt att kunden får tillit för en design, detta kan vara svårt att få fram genom en ritning på papper eller i två dimensioner på en skärm. Ytterligare en risk med en tvådimensionell prototyp är att den nödvändigtvis inte realiseras på det sätt som kunden hade förutspått och således blir en besvikelse när kunden kliver in i den. [68]

När VR kom in i bilden kunde arkitekter och dennes klient ta visualiseringen av bygget till en ny nivå. VR låter kunden kliva in i konstruktionen och faktiskt känna hur den skulle vara färdigbyggd, VR ger en upplevelse som inte går att matcha med traditionella 2D bilder eller sketcher. VR ger också designers möjligheten att arbeta kollaborativt på en ny nivå, utvärdera designbeslut tillsammans i realtid, ljusets påverkan på konstruktionen för olika tider på dygnet och så vidare [68].

Teknologin har minskat tiden och kostnaden det tar att skapa byggnadskonstruktioner med hänsyn på den flexibilitet den erbjuder designers, den har givit arkitekter möjligheten att ta viktiga beslut i designprocessen i realtid. Denna process kallas för *Digital Prototyping* där man använder sig av en digital miljö för att skapa sin konstruktion. Processen erbjuder en chans att hitta fel redan i designprocessen innan en realisering av konstruktionen har påbörjats och på så sätt minskar kostnader och fel som skulle kunna uppkomma i byggandet. [71]

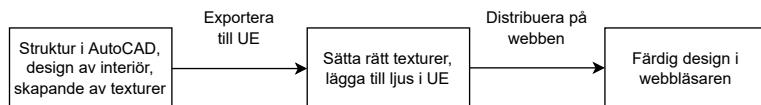
C.4.3 Att visualisera konstruktioner i UE & VR

För att ta reda på hur visualiseringar fungerar i UE & VR har en studie analyserats om hur designen av ett hus kan visualiseras. Ytterligare en studie har använts, denna gäller simu-

lering av ett fordon som gjorts för att ta reda på hur olika faktorer påverkar såsom extrema väderförhållanden och vägunderlag.

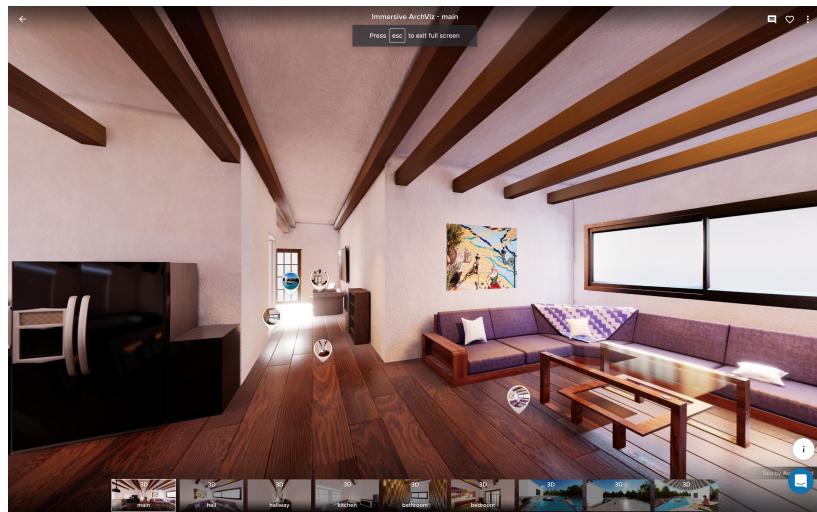
Visualisering av ett hus

En stor utmaning för arkitekter som nämndes i avsnitt C.4.2 var att sälja in ett projekt till en kund. Att presentera projektet som försöker säljas blir lättare om det går att se det färdigbyggt, detta är förstår inte alltid möjligt då byggnadsprojekt säljs först innan det byggs. I en studie av Arvin m.fl (2022) [68] har en design gjorts i ett CAD-program som heter *Autocad*, där har strukturen av huset uppförts. Modellering av interiören gjordes i olika 3D-program. Texturer gjordes med hjälp av mjukvaran *Substance Painter*. Hur processen gick till steg för steg går att se i figur C.1. [68]



Figur C.1: Diagram på processen för visualisering i UE.

När konstruktionen var färdig i UE distribuerades prototypen på en webbsida. Utan ett VR-headset är det möjligt att utforska byggnadens design i en 360-graders panoramautsikt. Med hjälp WebVR går det även att se den med ett VR-headset. En bild på den sluttgiltiga designen går att se i figur C.2 [72].



Figur C.2: Visualisering av bostad.

Fordonsmodellering i UE

I en rapport utforskar Sapienza m.fl (2021) [73] huruvida UE lämpar sig för att testa designen på fordon. Detta görs genom att ändra miljön såsom väderförhållanden och vägunderlag. Processen var till en början lik den förra för visualiseringen av ett hus, det grafiska gjordes utanför UE och importerades. Fokuset i denna rapport låg förstår på det fysikaliska resultatet av testerna och mindre på hur designen var visuellt.

Den simulation som gjordes var att samma rutt användes i olika väder- och vägförhållanden, sedan jämfördes rutterna. Dessa tester gjordes på två olika militära fordon för att bättre

analysera hur dessa fungerar i olika former av terräng. Det problem rapportskrivarna hade var att få fysiken att fungera som den bör, det som är förinställt i UE är inte nog bra för verklighetstrognna simulationer. Det finns oerhört många parametrar att justera, resultatet var någorlunda verklighetstrognna simulationer med utrymme för förbättringar i framtiden. [73]

C.4.4 Ett verktyg för visualisering och samarbete

Då tidigare resultat i avsnitt C.4.1 påvisat att traditionellt arbete i CAD kan föra med sig negativa konsekvenser finns ett tydligt behov av ett verktyg som erbjuder användaren en möjlighet att arbeta tillsammans och inlevelsefullt. I den rapport som analyserats för att skriva detta stycke har ett verktyg för visualisering och samarbete tagits fram.

Innan författarna satte igång med utvecklandet intervjuades ett antal anställda inom byggsbransen, allt från juniora till seniora medarbetare, chefer till designers. Syftet med intervjuerna var att finna de exakta behov som finns för ett sådant verktyg som ska utvecklas.

Behoven delades upp i kategorier, dessa var följande, med antalet krav i kategorin i parentes:

- Generella (2)
- Visualisering (7)
- Kommunikation (5)
- Åtkomstkontroll och crowdsourcing (5)
- Speciella behov (2)

Exempel på behov är allt från inlevelse och en möjlighet att gå runt i miljön till kommunikation i programmet genom röst och chatt. Den slutgiltiga produkten hade alla dessa funktioner och fler därtill med några nämnvärda som en möjlighet att rita på ytor och i luften, peka med ett inbyggt pekdon, manipulera objekt och ett inbyggt röstsysteem för att teamet ska kunna ta beslut direkt i systemet.

Resultatet av studien var dels mjukvara som snabbt kunde tillfredsställa många av de krav som togs fram i intervjuerna men också insikter om att virtuella designmöten med kommunikation har möjligheten förbättra design i byggarbeten som i sin tur leder till bättre byggen. [70]

C.4.5 Ett steg till - haptisk feedback

Design i en 3D-miljö och med integrationen av VR är stora steg som tagits inom designindustrin men för att ta ännu ett steg i utvecklingen har en rapport som handlar om att inkludera haptisk feedback i allt detta analyserats. I rapporten har ett haptiskt gränssnitt med hjälp av en handske skapats för att utöka inlevelsen och ge designers en faktisk fingertoppskänsla i VR-miljö. Syftet med den analyserade rapporten är att utforska VR:s teknologiska begränsningar men även de begränsningar som kan finnas i designdrivna miljöer där handspårning och igenkänning av gester behövs i realtid.

De kontroller som används för att interagera i en VR-miljö kan vara svåra att hantera till en början och inte ge användaren en möjlighet att vara precisionssäker utan den fokuserar enbart på att ”sikta och klicka”. Dessa verktyg i en sådan miljö ger användaren en möjlighet att endast utnyttja VR-miljön som en grafisk miljö och inte som en miljö där något faktiskt formges. Med en handske med haptisk feedback får användaren en möjlighet att designa som att det som ses i den virtuella världen hade varit i verkligheten.

För att tillverka handsken behövde olika värden mäts, dessa var alltfrån avstånd mellan fingrar, omkrets på fingrar, finger-, arm- och handplacering, vilken gest som gjordes, antal fingrar som pekar och så vidare. Denna data användes i Unreal Engine för att sedan skicka signaler till en Arduino mikrokontroller som i sin tur skickade feedback till handsken, där själva haptiken sker.

Det slutgiltiga resultatet av rapporten visar att en användning av haptisk teknologi i kombination med en virtuell miljö kan gynna resultaten för en designverksamhet. Resultaten visar även på att nivån som denna haptiska teknologi ska ligga på är för tillfället vag och mer forskning behöver göras för att få svar på detta. [65]

C.5 Diskussion

I denna del av rapporten diskuteras metoden och det som har framkommit i resultatet.

C.5.1 Metod

Då den metod som använts förlitat sig helt på vetenskapliga texter genom en litteraturstudie behöver inte resultatet nödvändigtvis spegla hur det ser ut på arbetsplatser idag, utan lutar mer åt vad som forskas på för tillfället. För att få en bättre översikt på hur det ser ut på arbetsplatser skulle en fältstudie på arbetsplatser kunnat göras eller intervjuer med folk i branschen. Problemet med denna metod dock är att en av frågeställningarna frågar vilka branscher som använder sig av spelmotorer och VR, därför hade denna metod eventuellt haft för stor påverkan på resultatet om ett för snävt antal branscher besökts.

C.5.2 Resultat

Med val av metod följer ett resultat som ska användas för att svara på frågeställningarna. Då metoden varit baserad på vetenskapliga uppsatser kan det vara svårt att dra slutsatser till den tredje frågeställningen eftersom den grundar sig i hur det ser ut i arbetslivet och inte i den akademiska världen. Trots detta går det att argumentera för att spelmotorer används i branscher då resultaten visat på att det är mycket användbara verktyg och samtidigt inte alltför nya vilket betyder att metoderna redan bör ha anammats.

Det går dock att argumentera för att resultatet har varit framgångsrikt för att svara på frågeställning ett och två då dessa går in på faktiska processer med slutsatser som fungerar i båda världar, den akademiska världen och i yrkeslivet.

C.6 Slutsatser

I detta avsnitt besvaras de frågeställningar som ställdes i början av rapporten.

C.6.1 På vilka sätt kan spelmotorer förändra designprocessen?

Det är tydligt att användandet av spelmotorer i en designprocess har påverkan. I en spelmotor är det enkelt för designers att snabbt påverka miljön, exempelvis faktorer som ljuskällor och tid på dygnet. Det är även enkelt i en spelmotor att göra små förändringar i sin design för att på så sätt testa olika variationer av samma design. En spelmotor utökar även möjligheterna för kollaboration mellan både kund och de som arbetar på designen i och med flerspelarfunktioner som ofta är inbyggda i spelmotorer.

C.6.2 Hur kan VR användas för att skapa en mer interaktiv designprocess?

En designprocess i kombination med VR låter användaren vandra runt i en miljö som skulle efterlikna den verkliga. I den virtuella miljön är det lättare att få en känsla av hur konstruktio-

nen skulle vara och ger användaren möjlighet att ändra på saker direkt i miljön med en direkt referens till hur det skulle vara i verkligheten. Mjukvara skapad för VR och för kollaboration i en designmiljö öppnar också upp för otroliga möjligheter i en designprocess för både designers och för kunder då man har möjlighet att både kommunicera och påverka miljön runtom sig i realtid.

C.6.3 Vilka branscher använder sig av spelmotorer som Unreal Engine i sina designprocesser?

Som nämntes i avsnitt C.5.2 är det svårt att svara på frågan utifrån resultatet. Med den metod och de sökkriterier som valdes var det främst rapporter relaterade till bygg- och arkitekturbranschen som visades som resultat även om sökkriterierna var generella och inte specifikt riktade till denna bransch. Det går att dra en slutsats, om än svag, att det är förekommande i denna bransch. För att ta reda på om det används i fler branscher skulle mer forskning behöva göras, exempelvis med den metod som nämns i avsnitt C.5.1. I kapitel B går det att läsa mer om branschers tillämpningar av Unreal Engine.



D

Metodik vid framtagning av krav

- Tony Lindbom

D.1 Introduktion

Detta kapitel kommer handla om vad det finns för metoder att få fram krav om det inte finns någon kund att arbeta mot. Många projekt börjar oftast med en idé eller en kund som har en. Analysansvarig skulle i vårt fall tillsammans med kunden och med hjälp av diskussion, få fram vad kundens behov för produkten var. Efter ett möte med kunden diskuterades önskemål och krav på funktionalitet och från detta skapades en kravlista. Dessa krav bör uppfyllas och ha olika prioritet för att man i utvecklingssynpunkt ska veta vad som skall utvecklas först, kraven ska ligga till grund för hur produkten ska byggas.

D.1.1 Syfte

Syftet med denna rapport är att undersöka och visa på olika sätt att få fram krav och jämföra dem mot varandra.

D.1.2 Frågeställningar

I den här rapporten ställs tre frågor med fokus på kravframställning och deras metoder som listas nedan. Dessa frågor kommer behandlas under rapportens senare kapitel.

1. Vilka generella kriterier finns det för att en produktidé ska ge värde för en kund?
2. Vilka metoder kan användas för att ta fram krav för ett mindre projekt?
3. Vilka fördelar och nackdelar finns det med de olika metoderna?

D.2 Teori

Detta kapitel kommer ta upp den teori som berör frågeställningarna.

D.2.1 Kundnöjdhet

I artikeln *A thorough literature review of customer satisfaction definition, factors affecting customer satisfaction and measuring customer satisfaction* av Razafimanjary Maminaina Aimee står det om faktorer som spelar in för att uppnå hög kundnöjdhet. Kundnöjdhet är baserat på de faktorer som har betydelse för att en produkt eller system ska ge värde för en kund. Dessa faktorer skiljer sig självklart från industri till industri. Men det finns vissa riktlinjer med faktorer som är generella för alla.[74]

1. *Produktkvalitet* - Kunden blir nöjd om den känner att produkten håller en hög kvalitet.
2. *Service kvalitet* - Kunden känner sig nöjd när produkten ger bättre service eller lika mycket som de förväntade sig utav den.
3. *Känslor* - Kunden känner sig nöjd för att produkten är av ett speciellt varumärke.
4. *Pris* - Produkter som har hög kvalitet men till ett rimligt pris kommer ha ett högre värde för kunden.
5. *Extra kostnader* - Om en produkt kan leveras utan extra kostnader eller att det inte behövs lägga ned tid för att få tillgång till produkten kommer höja värdet på den.

Produktkvalitet

Viktigast för att kunden ska bli nöjd ligger hos själva produkten och det är det som kan beskrivas som produktkvaliteten. I samma artikel av Razafimanjary Maminaina Aimee, står det att en av viktigaste egenskaperna hos en produkt är produktkvaliteten och den kan delas upp i sju delar.[74]

1. Produktens användningsprestanda
2. Egenskaper som extra funktionalitet
3. Att produkten är pålitlig och slutar inte fungera under produktens specificerade hållbarhetstid.
4. Att produkten håller vad som lovats i beskrivningen,
5. Att om produkten skulle gå sönder att den kan lagas snabbt, underhållas med lätthet och med låg kostnad.
6. Att estetiken på produkten (utseende, känsla, ljud o.s.v.) är bra, dock högst personligt.
7. Det sista är också en punkt som inte riktigt går att mäta och det är "känslan" av kvalitet.

Det finns mycket som gör att en kund kan känna värde för en produkt. Men själva känslan av värde ligger i största delen hos kunden själv.

D.2.2 Kravspecifikation

Är ett dokument som innehåller *krav*. Dessa krav ligger till grund för hur systemet skall fungera, vad det ska klara och vad det inte behöver klara. *Krav* definieras enligt IEEE 29148:2011E som "ett uttryck som förmedlar ett behov med dess begränsningar och villkor." [75]

När en *kravspecifikation* ska skrivas finns det en del frågor som ska besvaras.

1. *Funktionalitet* - Vad ska systemet göra?
2. *Externa gränssnitt* - Hur ska systemet interagera med människor, systemets hårdvara, annan hård- och mjukvara?
3. *Prestanda* - Vad är hastigheten, tillgängligheten, svarstiden, återhämtningstiden på de olika funktionerna i systemet?
4. *Utmärkande egenskaper* - Vad är portabilitetens, korrekthetens, underhållbarhetens, säkerhetens överväganden?

5. *Design begränsningar på en implementation* - Finns det någon bestämd standard att följa till exempel implementationsspråk, riktlinjer för databasintegritet, resursbegränsningar, utvecklingsmiljö?

Det finns saker som författaren av *kravspecifikationen* måste tänka på, till exempel att inte beskriva några designbeslut eller implementationsdetaljer för det finns andra dokument som skall ta upp dem.[76]

D.2.3 Marknadsanalys

Marknadsanalays är en metod för att undersöka och jämföra marknaden inför utveckling eller lansering av en ny produkt [27]. Detta genom att skaffa sig kunskap om hur marknaden ser ut, vad det finns för konkurrenter och vad den troliga efterfrågan av produkten skulle vara.

Idé

När ett nytt projekt påbörjas kommer det troligtvis ha en fas där idéer bollas om hur produkten ska se ut och fungera. I denna fas ska det finnas en grundidé att arbeta med. Denna grundidé kan vara en prototyp för projektet.[77]

Marknaden

För att ha en bra grund att stå på när utveckling av en produkt sker, är det bra att känna till marknaden. Efter att idén är påbörjad undersöks marknaden. *Upphandlingsmyndigheten* ger information och stöd för handel inom offentliga affärer. På den hemsidan finns det information om regler, lagstiftning, strategier, metoder och mall för marknadsanalays. Det finns flera olika sätt att göra det på men nedan kommer några exempel från *Upphandlingsmyndigheten*.[27]

- Mässor
- Intervjuer
- Enkäter
- Undersöka på internet
- Prata med troliga kunder/företag

Konkurrenter

Marknaden undersöks och det letas efter likande produkter och företag som kan ses ha en produkt som är konkurrerande. Det som undersöks är vad de har till skillnad mot den idén som tagits fram. Saker att titta efter kan ses i listan nedan.[27]

- Vad finns på marknaden idag?
- Vad kan kan produkten göra bättre?
- Vad skiljer sig som man kan göra vinst på?
- Kan produkten uppfylla det behov som marknaden har?

Kunder

I rapporten *Marknadsaktiviteter under en innovationsprocess* av Arne Johansson skrivs det om de olika aktiviteterna som sker från att en idé tagits fram och hur den ska lanseras på marknaden. I rapporten står det att när en produkt utvecklas det viktigt att veta vilken den tänkta kunden kommer vara och vilken typ av målgrupp som den ska rikta in sig på för att kunna implementera och lansera den så bra som möjligt.[77]

Analys

Efter att information samlats in analyseras den och därefter dras slutsatser från informationen. Med hjälp av analysen sätts de krav som skall ligga till grund för projektet.[77]

D.2.4 Konversationsmetoder

Konversationsmetoder är sätt att diskutera sig fram till det som ska stå i *kravspecifikationen*. I rapporterna *Requirements engineering: elicitation techniques* av *Sai Ganesh Gunda*, *En studie om användbarhetskrav: hur valet av insamlingsteknik kan påverka identifieringen av olika aspekter av användbarhet* av *Marcus Johansson* och *Techniques for requirements elicitation* av *Joseph A. Goguen och Charlotte Linde* kan man läsa om olika metoder för hur krav kan framställas i dessa rapporter tas också för- och nackdelar upp för de olika metoderna.[78, 79, 80]

Intervjuer

Intervjuer är enligt många den vanligaste metoden för att ta fram krav till mjukvaruutveckling, det finns flera typer av intervjuer men de två vanligaste och mest använda är *öppna intervjuer* och *stängda intervjuer*. Intervjuer sker en och en eller i gruppmöten där en person leder intervjun och ställer frågor som en eller flera personer ska svara och på sådant sätt få fram krav. Dessa intervjuer bör utföras av en som är kunnig på området och kan ställa sådana frågor som leder till bra svar som kan dras nytta av.

Stängd intervju är en intervjuform där det finns bestämda frågor som den som håller i intervjun vill ha svar på och samtalet kommer ledas runt dessa frågor för att försöka få fram svar på många av dem som möjligt av dessa frågor.

Öppen intervju kommer istället leda diskussion runt ämnet och frågorna kommer att ställas baserat på vart diskussionen går. I denna form går det mer ut på att försöka lista ut vad personen man intervjuar förväntar sig av produkten som skall utvecklas.

Idéverkstäder och fokusgrupper

Idevärkstäder och fokusgrupper är ofta större grupper med mäniskor som skulle kunna liknas med en gruppintervju, som tillsammans diskuterar fram idéer till krav. Dessa grupper har ett ”ledigare” tillvägagångssätt där frågor ställs till deltagarna. Idéer ska få komma fram naturligt och utan kritik från andra. Efter det tar man fram krav från dessa. Idéverkstäder kan leda till många krav som är svåra att realisera då metoden inte sätter några större gränser på idéerna. Fokusgruppen är väldigt lik Idéverkstäderna på många sätt men det skall finnas en starkare struktur i vad som skall tas upp och vilka frågor som skall arbetas med.

Idéverkstäder eller idékläckning som det också kan heta, samlar en grupp personer och med denna gruppen hålls en diskussion likt en *öppen intervju*. Där får idéer och tankar flöda fritt medan den som leder gruppen och tar anteckning om det som sägs, det är och den personens uppgift att repetera det som sägs och försöka arbeta vidare på den informationen som kommer fram.

Fokusgrupper är en fokuserad grupp personer som har ett tydligare mål med hela diskussionen, detta likt en *stängd intervju* fast för flera personer samtidigt. Gruppen kan bestå av till exempel experter inom området eller av utvecklarna själva. Det finns en tydligare agenda om vad som skall diskuteras och med hjälp av deltagarna skall det identifieras möjliga problem med produkten och sedan arbetas det fram lösningar för dem. I denna grupp diskuteras det fram krav tillsammans och gärna varför det ena kravet är bättre en det andra.

Brainstorming

Brainstorming är en mindre eller större grupp som ”ledigt” och öppet pratar om kraven och idéer får flöda fritt. Metoden används också när det finns ett specifikt problem som skall arbetas med som krävs att flera personer med olika kunskaper arbetar på problemet. Det finns oftast någon form av anteckning möjlighet för alla medlemmar eller en gemensam anteckningsyta där idéerna och lösningar skrivs ner. I dessa möten sätter man sig ner som grupp för att komma fram till lösningar på de problemen som ligger på mötets agenda. Från rapporten av *Sai Ganesh Gunda* tas det upp att med hjälp av brainstorming finns det vissa typer

av frågor som lämpar sig väl, nedan kommer en lista på de vanligaste problemen som kan diskuteras.

- Vad systemet borde förse för att lösa kundens behov?
- Vilka företags och branchregler måste produkten följa?
- Vad för frågor som skall ställas på intervjuer och som skall stå på frågeformulär?
- Vad finns det för risker med produkten och hur kan man undvika dem?

D.2.5 Frågeformulär

Enligt rapporterna av *Sai Ganesh Gunda* och *Marcus Johansson* är frågeformulär en välanvänd metod för att få statistik data. Frågeformulär når ut till en större mängd människor jämfört med de andra metoderna. Med hjälp av dessa formulär kan man få en bred grund till krav eller frågor som kan leda diskussioner om krav. Men det finns saker att tänka på när utformningen av dessa ska göras och det första är "*effektiviteten hos frågorna*", kommer formuläret att ge svar på det som du frågar? Det andra är "*ärligheten hos den som svarar*", kan du lita på att den som svarar, svarar ärligt utefter sin egen förmåga? [78, 80]

D.2.6 Analytiska metoder

Enligt rapporten *En studie om användbarhetskrav: hur valet av insamlingsteknik kan påverka identifieringen av olika aspekter av användbarhet* av *Marcus Johansson* är de analytiska metoderna "omvänt ingenjörkonst", vilket betyder att inför den nya produkten, går igenom dokument som redan finns och undersöker om det går att implementera de kraven på det nya produkten.[80]

Kravåteranvändning

Kravåteranvändning går ut på att undersöka ett redan existerande projekt som liknar det nya eller där det nya skulle kunna vara en vidareutveckling till ett annat projekt där det kan ärvä krav. Nya projekt och produkter ärver mer än hälften av alla sina krav från liknande produkter eller från tidigare produktversioner[78]. Det som är viktigt med de ärvda kraven är att undersöka att de verkligen fungerar som krav för just den nya produkten.

Dokumentationsstudier

Vid dokumentationsstudier undersöks och analyseras tillgängliga dokument till exempel rapporter, filer, formulär och manualer för att dra slutsatser om krav som kan fungera för projektet. Men eftersom det är en ny produkt kommer det troligtvis vilja utvecklas nya lösningar på problemen till skillnad från de som står i dessa dokument men de kan användas som referensmaterial. Det som också är viktigt för den som utför studien är också att bestämma om de kraven som hittas skall vara som de är eller om de skall formuleras om på något sätt. [81, 80]

D.2.7 Syntetiska metoder

Metoder där en simulering eller tidig prototyp av systemet utvecklas för att skaffa sig en inblick av kravens konsekvenser.

Prototyper

Att skapa en tidig prototyp kan vara effektivt för att få sig kunskap om vad krav kan ha för konsekvenser i systemet. Detta genom att testa i ett tidigt skede om vad som skulle fungera med grundidén eller om det måste ändra något. Sedan arbetas det med iterationer av denna prototyp för att ta fram fler krav under projektets gång. Prototyper fungerar väldigt bra när

användare ska interagera med systemet eller produkten för att testa funktionalitet. [81, 80, 82]

Scenariot

Scenariot är en metod där händelser målas upp för systemet och sedan arbetar sig de som ställs inför scenariot, igenom alla tänkta delsystem och gränssnitt som kommer påverkas av det. Det målas ofta upp flera olika scenariot för att då få en variation på vad systemet ska och inte ska kunna göra.[81]

D.3 Metod

De metoder som kommer användas är en litteraturstudie och egna erfarenheter som dokumenterades under utvecklingen av produkten Smart City.

D.3.1 Litteraturstudie

Befintlig information på ämnet kommer undersökas och granskas för att sedan kunna jämföras mot den marknadsanalys som gjordes till *Smart City*. Med hjälp av litteraturstudien ska kunna komma fram med information för att svara på frågeställningarna. Sökmotorerna som användes var

- DiVA
- IEEE
- Google Scholar

Sökord som användes var "*requirements elicitation*", "*marknadsunderökning*", "*kravframställning*" och "*kravspecifikation*"

Urval av litteratur

Vid litteraturstudien skedde det ett urval bland den litteratur som hittades. Urvalet baserar sig på det som stod i abstraktet och resultatet samt titeln på rapporten. När sökningen gjordes användes de tio första sidorna som urval. Utöver detta användes också nyckelorden i de olika rapporterna för att se om rapporterna kommer innehålla det som letades efter mer specifikt.

1. Titeln på rapporten
2. Relevans till frågeställningen
3. Vilken bransch som berörs

Titeln på rapporten

Då det första som sågs var själva titeln på rapporten och efter ett par sökningar och genomläsningar, drogs slutsatsen att utifrån vad titeln sade kunde ett urval göras som gav relevans till denna rapport. De titlar som visade sig vara relevanta hade ord som *requirement elicitation*, *kravramtagning*, *metoder* och *kravhantering*.

Relevans till frågeställningen

På de sökorden som användes kom många rapporter fram där det inte gick att få fram tydlig information för att svara på frågeställningen. Därför gjordes urvalet att det skall finnas en relevans till frågeställningen.

Vilken bransch som berörs

Denna punkt valdes också med avseende på att med de sökorden som användes, kom det upp många artiklar och rapporter som berörde andra branscher för att ta fram krav och marknadsundersökningar. Dessa rapporter innehöll inte relevant information som man kunnat hoppas, därför gjorde urvalet att hålla sig till mjukvarubranchen.

D.4 Resultat

I det här kapitlet kommer det skrivas om det som framkom av litteraturstudien och egna erfarenheter.

D.4.1 Värde för kunden

Värde för en kund är väldigt subjektivt och kan ändras från produkt till produkt. Det syns en tydlig koppling mellan *produktkvalitet*, *kundnöjdhet* och *kundvärde* men det diskuteras fortfarande om exakt hur stor vikt de har för varandra. Men av de tre olika delarna är det produktkvaliteten som går att göra nånting åt när en idé utvecklas.[83]

När idén på en produkt är klar blir det då viktigt att tänka på de olika faktorerna som leder till kundnöjdhet som kan läsas om i D.2.1. När utvecklingen av idén sedan fortsätter är det den största delen *produktkvalitet* som innehåller många punkter som utvecklare kan arbeta med. Några av dessa punkter är estetik, egenskaper och funktionalitet.

D.4.2 Olika metoder för att ta fram krav

Det finns mängder av olika metoder för att ta fram krav till en *kravspecifikation*. I rapporterna *Effective requirements development: A comparison of requirements elicitation techniques* av Zheying Zhang, *Techniques for requirements elicitation* av Joseph A. Goguen och Charlotte Linde, *Requirements engineering: elicitation techniques* av Sai Ganesh Gunda och *En studie om användbarhetskrav: hur valet av insamlingsteknik kan påverka identifieringen av olika aspekter av användbarhet* av Marcus Johansson kan man läsa om de olika metoderna och hur de jämför dem mot varandra. Ofta används en kombination av flera metoder. Metoderna kan delas upp i olika kategorier som beskrivs nedan. Efter varje kategori nedan kan för- och nackdelar ses i en tabell för de mest använda metoderna för varje enskild kategori. [81, 79, 78, 80]

Konversationsmetoder

Dessa metoder samlar sig kring tanken att intervjuva enskilda personer eller grupper och genom en riktad konversation ska komma fram till de krav som måste uppfyllas. Intervjuerna kan vara *öppna* eller *stängda* vilket bestämmer hur strikt agendan för intervjun eller gruppen är. Frågorna till intervjun kan tas fram med hjälp av andra metoder till exempel ett frågeformulär. Intervjuerna hålls av en kunnig analytiker eller en person som har goda kunskaper inom området som produkten ska fylla ett behov. Konversationsmetoder kan vara tidskrävande beroende på hur många personer som intervjuas och dessa personer bör ha rätt färdigheter för att kunna svara på frågorna. som måste intervjuas för att få fram de krav som behövs för att utveckla produkten. Några exempel på dessa metoder är *intervjuer*, *tankeverkstäder*.[81, 79]

Tabell D.1: Tabell som visar för- och nackdelar för olika konversationsmetoder

Metod	Fördelar	Nackdelar
Intervju	+man får djup på kraven +man kan ställa väldigt specifika frågor +bra om när det finns en specifik kund	-är tidskrävande -krävs planering -kan bli svårt att få bredd på kraven
Tankeverkstad	+mer bredd på kraven på hög nivå +mindre tidskrävande	-krävs en del planering och förarbete -det kan krävas flera sessioner
Brainstorming	+bra sätt att få alternativa lösningar på krav +man får en bredd kraven +personerna som deltar kan ha olika kunskaper	-tar mycket tid -en del av idéerna kommer inte kunna användas -svårt att få alla att bli hörda

Observationsmetoder

Metoder som går ut på att observera mänskligt beteende och vanor genom att en person iakttar människor i deras miljö där produkten ska användas. Detta ger en rik förståelse för användningsområdet för produkten och lämpar sig väl för krav som är svåra att formulera med ord för en person. En av dessa metoders nackdelar är att det kan ta mycket längre tid jämfört med andra metoder, vilket skulle kunna vara avgörande för en produkt med väldigt kort tidsram för utvecklingen. Detta då en person ska följa med på arbetsplatser och observera hur personer arbetar och det kan behöva ske i många omgångar. Det som talar för dessa metoder är att man genom dessa observationer kan få svar på frågor som är svåra att formulera och det kan också ge en djupare förståelse för hur en produkt skulle kunna fungera i sin menade miljö. Exempel på observationsmetoder är *socialanalys* och *protokoll analys*. [81]

Tabell D.2: Tabell som visar för- och nackdelar för olika observationsmetoder

Metod	Fördelar	Nackdelar
Social analys	+bra för att finna krav som är svåra att formulera muntligt +man kan passivt samla in data och sedan analysera	-det är väldigt tidskrävande -krävs en nogrann person som kan analysera informationen
Protokollanalys	+användare pratar om sina tankeverkstäder och beskriver tankeprocessen +inblickar i krav om funktionositet och arbetsflöde	-måste tolka det som användaren säger -vill det uppnås bredd måste det finnas många olika uppgifter som användaren genomför

Analytiska metoder

Analytiska metoder används för att utvinna krav från en mängd olika typer av dokumentation. Dokumentationen kan vara andra *kravspecifikationer*, krav från äldre versioner av produkten eller liknande produkter. Andra användbara dokument är manualer och standarder som ska följas. Det som är bra med de analytiska metoderna är att om dokument finns kan en *kravspecifikation* snabbt skapas med dessa. Detta kan ses som en fördel om projektet har en snäv tidsram att följa och snabbt måste komma igång med utveckling. Det som skulle kunna ses som negativt är att gamla krav från äldre dokument måste arbetas om för att passa just det nya projektet. *Kravåteranvändning* och *dokumentationstudier* är några av de vanligaste analytiska metoderna.

Tabell D.3: Tabell som visar för- och nackdelar för olika analytiska metoder

Metod	Fördelar	Nackdelar
Kravåteranvändning	+kan ge en färdig grund till kravspecifikation +kostnadseffektiv +tidseffektivt	-kraven kanske inte passar för just denna produkten -krävs diskussion om kraven kan användas eller inte
Dokumentationsstudier	+lätt att nå ut till många personer +kostnadseffektiv +bra för att hitta krav som är omedvetna	-gamla krav kanske inte passar den nya produkten. -finns brister i metoden och bör kombineras med andra metoder

Syntetiska metoder

Oftast är målet att ta fram krav innan man utvecklar något för att man ska ha vissa riktlinjer, dock är vissa krav svåra att få fram utan att testa produkten eller viss funktionalitet [78]. Syntetiska metoder skapar ett *användascenario* eller en *prototyp* för att tidigt kunna testa en idé och på så sätt ta fram krav. Dessa metoder används främst när krav för användargränsnitt eller funktionalitet som användare behöver interagera med ska tas fram, alltså är *prototyper* och *användarscenarion* bra metoder för att uppnå god användbarhet. Det finns mycket som är bra med de syntetiska metoderna till exempel att det är flexibelt och man kan tidigt upptäcka fel med produkten som man annars inte skulle kunnat upptäckt försen i slutskedet av utvecklingen. Det som är negativt med dessa metoder är att det ofta är väldigt dyrt att ta fram prototyper och i de flesta fall måste krav redan existera för att kunna göra *scenarion* och *prototyper*. [80]

Tabell D.4: Tabell som visar för- och nackdelar för olika syntetiska metoder

Metod	Fördelar	Nackdelar
Användarscenarion	+kan skräddarsy för specifik del i systemet +det är flexibelt hur kraven utvinns. +fördjupning och specificerar av krav	-det borde finnas en grund till kraven -kräver att utvecklare deltar i framtagning av scenarien.
Prototyp	+man får krav som berör användargränsnitt +man kan tidigt upptäcka fel +man får större förståelse för produkten +man upptäcker lättare funktionalitet som saknas	-ofta dyrt att skapa prototyper -måste ha krav innan prototyp skapas som grund. -kan finnas för stor variation på kraven att det inte går att bygga en tidig prototyp

Marknadsanalys

En marknadsanalys går ut på att undersöka hur väl produkten som skall utvecklas skulle stå sig på marknaden. Det som undersöks är liknande produkter, konkurrenter, troliga kunder och målgrupper. Informationen samlas in från en specifik problemställningen inom marknadsområdet till exempel *det finns ett problem och en idé som kanske kan uppfylla det behovet på marknaden* och med den utgångspunkten undersöker man hur väl den idén skulle fungera. Efter det tas krav fram baserat på en analys av informationen från undersökningen. Det som är bra med en marknadsanalys är att få reda på hur marknaden ser ut och vad det finns för

liknande produkter på marknaden i dagsläget. Det som skulle kunna ses som negativt är att det krävs kompletterande metoder för att få ut ordentliga krav från själva analysen och detta skulle kunna leda till att det tar längre tid. [27, 77]

Tabell D.5: Tabell som visar för- och nackdelar hos marknadsanalys

Metod	Fördelar	Nackdelar
Marknadsanalys	+man får en klar bild om vad som finns på marknaden +finns många olika sätt att utföra den på	-ganska lång process med mycket undersökning och granskning -svårt att få ut tydliga krav

Frågeformulär

Frågeformulär är en metod som ger en bredd på kraven. Det är dock svårt att få djupet. Denna metod kan nå ut till många människor till låg kostnad på väldigt lite tid. Enligt rapporten av *Marcus Johansson* lämpar sig metoden bäst för två typer av situationer: "att samla statistisk data för att bekräfta en misstanke och att samlar åsikter eller förslag" [80]. Dessa formulär kan också hållas anonyma. Frågeformulär kan också användas som hjälpmittel eller komplement till många av de andra metoderna till exempel för att ta fram givande frågor till intervjuer och att tunna ut krav efter en marknadsanalys. [81, 78, 79]

Tabell D.6: Tabell som visar för- och nackdelar hos frågeformulär

Metod	Fördelar	Nackdelar
Frågeformulär	+lätt att nå ut till många personer +kostnadseffektiv +tidseffektivt +statistisk data	-svårt att få till ett bra formulär -svårt att tolka -personer som inte svarar eller svarar oärligt

D.5 Diskussion

I detta kapitel kommer det diskuteras om det som kom fram i resultatet.

Värde för en kund är ett begrepp som är väldigt subjektivt. Det finns många olika saker som kan ge värde för en kund men många av de sakerna har med produktkvalitet att göra [83]. För att riktigt kunna ta reda på vad som ger värde på en specifik marknad och produkt krävs mycket förundersökning och analys av det material som kommer fram. Några bra sätt att testa marknaden kan vara att skapa en prototyp som låter troliga kunder prova produkten. Med hjälp av detta kan feedback på nuvarande produktkvalitet, funktionalitet och önskemål om förbättringar eller ändringar fås. Detta ser ut att stämma med de artiklar och texter som togs fram med hjälp av litteraturstudien.

Välja metod kan vara svårt. I vårt fall gjorde vi en marknadsanalys och resultatet av den kan läsas i 5.1. Hade vi haft mer tid under projektets gång skulle vi tänka oss att använda olika metoder för att få en grundligare *kravspecifikation*. Då skulle vi kunna experimentera lite med olika metoder som skulle kunna bättre insikt i marknaden, ge bättre funktionalitet för systemet och kanske kunna få en bättre produkt i slutändan. De metoderna som skulle kunna ge best resultat för produkten SmartCity skulle troligtvis vara, *brainstorming*, *frågeformulär*, och någon form av *observationsmetod*. Eftersom projektets tidsram var ganska stram fanns det bara möjligheten att följa en metode och sedan diskutera fram krav runt det som kom fram.

Den metod som valdes för SmartCity var *Marknadsanalys*, för efter lite forskning om liknande produkter kunde man se att det fanns en marknad för just denna typen av produkt. Men som sagt skulle vi haft mer tid för projektet skulle vi kunna implementera fler metoder för att få en mer omfattande *kravspecifikation*.

Brainstorming skulle vara en bra start då alla medlemmarna i gruppen skulle sätta sig ner, bina ut idén ordentligt för att få en grov grund på de krav som alla i gruppen känner är relevanta. Alla skulle då också få flika in med egna idéer och lösningar på implementationer.

Frågeformulär skulle kunna vara nästa steg då man har en grund med krav som behöver vidareutvecklas eller om vi får ut så mycket krav att det måste avgränsas lite. Frågeformulären skulle då till liten kostnad kunna nå ut till en stor mängd människor som skulle kunna ”hjälpa till” att minska mängden av de kraven som finns. För det kan finnas ett visst överflöd av krav som är väldigt liknande varandra eller så finns det för lite krav och man behöver nya idéer eller kommentarer på det som man redan har kommit fram till.

En observationsmetod skulle i vårt fall vara en bra metod för att se hur miljöerna och marknaden som produkten är tänkt att fungera för. Detta för att få en klar bild på hur personer skulle arbeta med den och på sådant sätt kunna få direkt feedback på vad för funktionalitet som produkten skulle behöva. Detta leder också till att fler krav för produkten tas fram men med mer inriktning på hur den ser ut och fungerar.

Detta skulle då leda till att krav med djup, bredd och idéer på fler krav tas fram.

Egenskaper hos olika metoder: I tabellerna D.1 till D.6 sammanfattas de största för- och nackdelarna hos varje metod som tas upp i kapitel D.4.2. De största för- och nackdelarna hos de olika metoderna kan sammanfattas till *tid*, *förberedelser* och *vad för typ av krav* man får ut från respektive metod. Med hjälp av dessa egenskaper och den tidsram som projektet har att rätta sig till, ska kunna anpassa sig mer med de metoder som är mest lämpade för just det projektet.

Tid är för de flesta projekt den avgörande faktorn om vad man hinner utveckla och producera. Detta leder till om det finns möjlighet att välja metoder som skulle kunna spara in på tid för kravframtagning och fortfarande få fram giltiga och relevanta krav är det bra.

Förberedelser: Denna egenskap har en anknytning till tiden men att med dessa förberedelser kan tid besparas. Förberedelser har att göra med om del av produkten färdig för att kunna utnyttja sig av vissa metoder, de metoder som kräver förberedelse är bland annat *syntetiska metoderna* och *konversationsmetoderna*. Dessa kräver mer förarbete för att ta fram materialet som metoderna grundar sig på.

Typ av krav är en annan del av det som utvinns från de olika metoderna och man kommer behöva använda sig av olika metoder för att till exempel få krav om användargränssnitt, funktionalitet och krav som är svåra att formulera. Denna punkten kan tålas att tänka på, då med de andra två punkterna får med det omfånget av krav som projektet kräver.

D.6 Slutsatser

I det här kapitlet kommer det svaras på frågorna från frågeställningen.

Vilka generella kriterier finns det för att en idé ska ge värde för den fria marknaden?

En av de största faktorerna för att ge värde för en kund på den fria marknaden är att utveckla en produkt av hög kvalitet. En högre kvalité kan uppnås om utvecklingen av produkten sker efter det som står i D.2.1.

Vilka metoder kan användas för att ta fram krav för ett mindre projekt?

Det finns flera metoder för att ta fram krav till ett mindre projekt och de kan sammanfattas till *Konversationsmetoder, observationsmetoder, analytiska metoder, syntetiska metoder, marknadsanalys och frågeformulär*.

Vilka för- och nackdelar finns det med de olika metoderna?

Det fanns tre gemensamma faktorer bland för- och nackdelarna och de var, *tidsåtgång, förberedelser och vilka typer av krav* som kan utvinnas från en viss metod. En djupare lista på alla metoder som tas upp i rapporten och deras för- och nackdelar finns i tabellerna D.1 till D.6.



E

Hur uppmärksammar man en användare när det finns oändligt med information - Simon Magnusson

E.1 Introduktion

Ett sätt att se på Smart City är att jämföra det med ett flygtorn. Dataströmmar från flera håll strömmar in i ett enda stort flöde av information som användaren måste sålla mellan för att kunna få en helhetsöverblick. Händer något viktigt är det en prioritet att informera om att det hänt och att se till att det uppmärksammias. Att använda ett sådant system kräver att användaren är fokuserad. Detta ställer dock krav på systemet att också vara tillbakadraget och inte vara i vägen för användaren. Den kognitiva ansträngning som ett system utgör är en reell faktor i hur väl en användare kan åstadkomma vad de vill göra. I kritiska situationer är det svårt att veta hur en användare kommer använda ett system och det är ofta utom kontroll. Det som dock går att göra är att designa system som inte är i vägen för en användare genom att tänka igenom hur gränssnittet är utformat. Att se till att något uppmärksammias är ett svårt problem då alla människor fungerar olika, men vad finns det för gemensamma faktorer hos gemeneman som man kan nyttja för att uppnå detta?

E.1.1 Syfte

Syftet med denna rapport är att redogöra hur ett gränssnitt kan utformas för att dirigera en användares uppmärksamhet och redogöra vad som kan göras för att designa gränssnitt som är mindre kognitivt ansträngande för användaren. Denna rapport avser inte att utreda alla de psykologiska aspekter som knyts an med uppmärksamhet. För att ge en mer snäv precision av vad som kommer studeras är det en analys av gränssnittsdesignpraxis som möjliggör att användaren med stor sannolikhet blir informerad om allt relevant i användarens kontext utan att tappa fokus. Fokus i denna raports kontext innebär att ett systems gränssnitt inte utgör onödig belastning på användaren som konsekvens av gränssnittets utformning.

E.1.2 Frågeställningar

1. Hur kan man gå tillväga för att designa ett gränssnitt som är mindre kognitivt ansträngande?

-
2. Vad finns det för tillvägagångssätt för att påkalla uppmärksamhet hos en användare i ett gränssnitt?

E.2 Teori

Denna del tar upp teori som är relevant för rapporten.

E.2.1 NASA Task Load Index (NASA-TLX)

Detta är en metod framtagen av NASA för att utvärdera hur ansträngande ett system uppfattas vara att använda. Metoden går ut på att direkt efter en uppgift gjorts av en testperson får personen gradera upplevelsen genom frågor. Dessa frågor utgår från sex subjektiva delar: mental belastning, fysisk belastning, tidsmässiga krav, prestation, ansträngning och frustration. De frågor som ställs är följande:

1. Hur mentalt påfrestande var uppgiften?
2. Hur fysiskt påfrestande var uppgiften?
3. Hur bråttom kändes det att göra uppgiften?
4. Hur väl lyckades du genomföra uppgiften du blev bedd att göra?
5. Hur svårt var det att göra det du gjorde?
6. Hur osäker, irriterar eller stressad var du när du genomförde uppgiften?

Dessa frågor besvaras utifrån en 1-20 skala och därefter ställs 15 frågor som fokuserar på att rangordna de olika områdena efter signifikans. Ett exempel på en fråga är huruvida fysisk belastning är mer centralt i uppgiften gentemot mental ansträngning. Efter dessa frågor viktas frågorna utefter signifikans och ett svar på en 1-100 skala kan utvinnas. Ju högre siffra på denna skala desto mer kognitivt ansträngande antas systemet vara för användaren. Denna metod har blivit citerad i mer än 4400 studier vilket belyser den stora inverkan metoden har haft på att utvärdera system. [84, 85, 86]

E.3 Metod

Denna sektion redovisar den metod som kommer användas för att genomföra denna analys.

E.3.1 Metodval

Den metod som valdes för att besvara frågeställningarna var att genomföra en litteraturstudie. Detta innebär att den information som tas fram är från kvalitetsgranskade källor som bidrar med information som kan besvara frågeställningarna.

Den sökmotor som används för att hitta källor är Google Scholar. Med hjälp av denna sökmotor har ett antal söktermer använts, dessa listas nedan:

1. User interface
2. User interface attention
3. NASA-TLX user interface
4. NASA-TLX attention

Urvalet av källor baserades på att snabbt filtrera bort källor baserat på titel. Hittades artiklar som kunde vara av relevans för ämnet lästes sammanfattningen och diskussionen för att få mer information gällande relevansen till frågeställningarna. De rapporter som valdes ut och användes i rapporten har valts utifrån dess relevans till att besvara frågeställningarna.

E.4 Resultat

Denna del går över resultatet från litteraturstudien.

E.4.1 Layout

Att bygga upp ett komplext gränssnitt är inte trivialt. Det finns många olika tillvägagångssätt till att sätta upp en layout för ett system med många komponenter. Exempel på ett sådant system är mjukvaran som används för sjukhusjournaler. I en studie från *Mayo Clinic College of Medicine* [87] evaluerades två olika sjukhusjournalgränssnitt som användes i intensivvårddelningar för att se vilken inverkan olika gränssnitt har på en användares prestation. Två olika gränssnitt evaluerades, det nuvarande gränssnittet som sjukhuset använde och ett omesignat gränssnitt. Skillnaden mellan gränssnitten var antalet datapunkter som presenterades och strukturen av visningen. Standardgränssnittet visade 1007 datapunkter medan det omesignade gränssnittet visade 106 datapunkter. Det nya gränssnittet hade skapats utifrån att endast visa det mest relevanta i en intensivvårddelning men också gruppera information baserat på korrelation. Information som ansågs överflödig lades bakom menyer. Dessa två gränssnitt utvärderades genom att testpersoner fick genomföra olika uppgifter med hjälp av ett av gränssnitten och sedan utvärdera genom NASA-TLX metoden. Standardgränssnittet gav en genomsnittligt 58 poäng utefter NASA-TLX-skalan och det omesignade gränssnittet gav 38.8 poäng. Detta tyder på att testpersonerna upplevde att det nya systemet var mindre ansträngande att handskas med. Detta indikerades även genom att tiden det tog att genomföra de förutbestämda uppgifterna minskade också när testpersonerna fick använda det nya gränssnittet. [87]

Liknande resultat har hittats när designen på två typer av kärnkraftverkskontrollers utvärderats. En simulerad miljö skapades med hjälp av en befintlig applikation för att hantera vattenpumpar på ett kärnkraftverk. Orginalversionen av denna applikation testades och jämfördes med en omesignad version av applikationen där modifikationer till layouten över systemet hade skett. De övergripande modifikationerna som gjordes var att skymma undan viss data som ansågs överflödig, justerade sådan att grupper av komponenter låg placerade i linje, etc. Testpersoner fick använda dessa två system, genomföra olika uppgifter och sedan utvärdera systemen utefter NASA-TLX-metoden. Resultaten visade på att det omesignade gränssnittet gav bättre värden i NASA-TLX-skalan och att den genomsnittliga tiden att genomföra uppgifterna minskade. Studien dokumenterade att den genomsnittliga tiden gick från 275.6s (+ 96.8s) med original gränssnittet till 231.9s (+ 81.4.s) för det nya. [86]

Layouten på gränssnitt har också en korrelation med hur många fel en användare gör. I båda studierna genererade de nya gränssnitten färre fel än vad orginalgränssnitten gjorde. [87, 86]

E.4.2 Animation

En användare har många sätt att utvinna information från ett system. Ett sätt som en användare kan göra detta är genom animation där knappar och iconer kan förmedla information genom rörelse istället för genom text. I en studie genomfördes tester för att se huruvida användare tolkar olika typer av animationer applicerade på iconer. Animationerna definierades som kinetiska då de endast ändrade ikonen fysiskt genom exempelvis studsningar, ut- och in-dragningsar, krossat glas, mm. Vissa animationer var starkt inspirerade av MacOSX anima-

tioner som hur ikoner skakar för att indikera att de kan flyttas. 39 stycken olika animationer togs fram som skulle representera 8 stycken olika typer av händelser.

Dessa händelser var:

1. Objektet genomför ett arbete
2. Objektet har bytts ut mot ett nytt
3. Objektet kan ej genomföra användarens förfrågan
4. Objektet är nytillagt
5. Objektet är inte längre tillgängligt
6. Objektet tillkallar uppmärksamhet
7. Objektet öppnas eller laddar
8. Objektet är flyttbart

Utifrån dessa kategorier fick testare se ett slumpräglat urval av tio animationer utav de 39 möjliga. Vid varje video fick testarna ge sin tolkning av vad animationen försökte framföra utifrån de åtta möjliga tolkningarna. Studien fann var att animationerna lyckades ta fram tolkningar utefter de som animationerna var framtagna efter.

En tolkning som blev vald av flest testare var tolkningen om att objekt tillkallar uppmärksamhet. Studien attriburerar detta till att rörelse ofta i sig är uppmärksammende. Studien ponderar möjligheten till olika grader av hur mycket något visuellt vill förmedla att något behöver uppmärksamhet. Animationer skapade för andra syften blev ofta tolkade som att de tillhörde tolkning nummer sex. Detta överlapp i tolkningar kunde inte endast ses för tolkning nummer sex. Ett exempel är att tolkningar om att ett objekt inte kunde genomföra en förfrågan och att ett objekt inte längre är tillgängliga. Dessa animationer kunde blandas ihop då animationer designade för vardera visade på att de kunde tolkas lika väl för båda. Studien attriburerar detta till att kontext är viktigt för att tolkningen ska bli korrekt. Det understryks att denna vetskaps om att samma animation kan tolkas varieras för olika kontexter ska tas till hänsyn när ett gränssnitt designas. [88]

E.4.3 Auditiva varningar

Något som kan komplementera det visuella är ljud. I en doktorsavhandling av Ulfvengren vid Kungliga Tekniska Högskolan (2003) studeras hur ljud uppfattas och uppmärksamas för att framställa karaktärsdrag kring vad som utgör en bra auditiv varning. Avhandlingen utgår från ett perspektiv gällande varningssignaler och hur piloter uppfattar dem. I texten definieras uppmärksamhet som ett filter. Ett filter som bestämmer vad för information som ska behandlas och vad som ska göras. Vad som uppmärksamas baseras på *informationskostnaden*. Denna "kostnad" är ett mått på hur mycket det krävs att hantera informationen och baserat på detta görs val om vad som får uppmärksamheten. Ljud är de sinne som aldrig stängs av. Hörselsinet tar oavbrutet in signaler och hjärnan processar denna information.

"Ljud verkar vara de sinne som har obligatorisk åtkomst till medvetandet, även om uppmärksamheten är vänt någon annanstans, ljud spelas in och processeras (även till rudimentär nivå) av hjärnan" [89]

I avhandlingen tas ett argument upp om att faktumet till att naturen inte försåg människan med öronlock grundar sig i att den akustiska kanalen är väl lämpad för auditiva varningar. Hörselns fungerar i alla riktningar. Ett ljud kan uppmärksamas även om en användare inte aktivt lyssnade efter det ljudet. Ljud uppfattas också snabbare i jämförelse med visuella varningar. Auditiva varningar visar sig få användare att reagera snabbare. Kombinering av visuellt och auditiva varningar snabbade inte upp responsen från användare. Avhandlingen tar upp att det är viktigt att implementera ljud i komplexa system för att öka effektivitet och, i kritiska situationer, öka säkerheten. Dock krävs det att ljuden är designande utefter

människan. Ett misstag som ofta görs är att ljuden är utformade på ett sätt som slösar på användares kognitiva resurser. För att minimera detta härleder avhandlingen egenskaper som är önskvärda hos en varningssignal:

- Har en naturlig innehörd i användarens kontext
- Är kompatibel med auditiv informationsprocessering
- Är behaglig att lyssna på (inte störande)
- Är enkla att lära sig
- Är effektiva i studen
- Minskar tiden att genomföra det som behövs göras
- Går att tydligt höra
- Går att urskilja från andra typer av varningar
- Går att urskilja från andra individuella varningar

Att utforma en varning handlar om att ge information till användaren. Ljud som ger information om varningen kan sänka *informationskostnaden* då det dämpar anledningen att visuellt förstå problemet. Ett argument görs om att varningar i regel ofta är utformade som enstaka toner. I naturen är det högst otroligt att örat blivit stimulerat med endast en frekvens. Öron har utvecklats för att plocka upp mönster av frekvenser och att förmågan att särskilja mellan olika mönster av frekvenser. När vi hör ett ljud analyseras ofta inte ton, rytm, klangfärg utan det som fokuseras är att finna ursprunget till ljudet. Avhandlingen gör ett exempel om att när ett glas med vatten fylls är det inte tonen av ljudet som får oss att förstå när det är fullt, utan det är ljudet av fullhet. En människa kan lära sig hundratals ljud men abstrakta ljud, såsom enstaka frekvenser, är inte inkluderat. Detta kan översättas till att det inte spelar någon roll huruvida många ljud som används utan att det handlar om associationerna som ljuden ger. Ett exempel fanns att om de traditionella varningsljuden i en bil byttes ut mot djurljud underlättade det för användaren när de skulle lära sig vad de betyder. Resurserna som går till uppmärksamhet är begränsade. När denna tröskel överskrids går prestationen ner. Att lära sig ett nytt ljud eller att komma ihåg ett ljud tar längre tid än att förstå vad ett ljud betyder.

En metod för att skapa bra ljud är att låta testpersoner som ska använda systemet vara med i designprocessen för att para ihop ljud med vad testpersonen tyckte att de varmades om. Sen efter de ljuden parades ihop fick testpersonerna säga hur mycket de tyckte de ljudet passade in. Utefter de modifierades ljuden och testen gjordes om. Denna metod hjälpte då varningssignalerna som skapades kunde reducera den kognitiva ansträngning som skedde när de användes vid flygning. [90]

E.5 Diskussion

I detta kapitel diskuteras resultatet och metoden som använts för att få fram resultatet.

E.5.1 Metod

Metoden förlitade sig på vetenskapliga artiklar för att ta fram resultatet. En aspekt som är svår att studera är hur intuition passar in i hur ett gränssnitt utformar sig. De artiklar som bearbetats i resultatet tangerar mycket på detta men det är sällan konkret fokus på de området. Att mer grundligt förstå detta skulle underlättas genom att genomföra intervjuer med folk inom olika områden om hur de känner kring de system som de använder. Ett problem med denna metod är att frågeställningarna utgår ifrån ett ämne som är väldigt brett. Det som skrivits i resultatet är inte uttömmande men nog med information anses lagts fram för att kunna dra slutsatser.

E.5.2 Kognitiv ansträngning

En tanke som dyker upp ur all information som presenterats i resultatdelen handlar om friktionslösa gränssitt. Att läsa något kräver ansträngning, likväl byta fokus från en sak till en annan. Gränssnitt kan vara utformade på ett sådant sätt att de är förståeliga men som kräver mer fokus för att genomföra en uppgift. Det kunde tydligt ses i kärnkraftverkskontrollertesterna där layouten i ett program visades ha en stor påverkan på testpersoners förmåga att göra arbetsuppgifter. Relativt enkla förändringar i den struktur som programmet har ger stor inverkan på användaren. Likväl i programmen riktade åt intensivvården där undanskymning av data som inte alltid är behövlig och istället lägga dem bakom lättillgängliga menyer ger luft åt gränssnittet som gör att användaren inte behöver sålla bland lika mycket information på en gång. Att gruppera information mer genomblickt, tillrättalägga gränssnittskomponenter eller använda animation för att förmedla information är bara några exempel på vad som kan göra skillnad mellan ett kognitivt ansträngande gränssnitt eller inte.

E.5.3 Associativitet

Associativitet är något som genomsyrar alla delar av resultatet. Om layouten i ett program grupperar information som hör samman på ett genomblickt sätt gör det att en användare inte behöver mentalt gräva efter lika mycket animation. Att använda animation kan minska friktion i gränssnittet genom att byta ut text mot rörelse som användare intuitivt redan förstår. Ljud kan bärta förmågan att minska nödvändigheten att processera något visuellt. Att tänka på associativitet gynnar då den begränsade uppmärksamheten användare har kan nyttjas åt att genomföra uppgifter istället för att belasta dem genom dåligt genomblickt gränssnitt.

E.5.4 Är något av detta applicerbart i Smart City?

En tanke som hör ihop med associativitet är att desto fler informationskällor som implementeras desto mer ökar målet för användaren att förstå datan. Att automatiskt gruppera information på kartan, eller verktyg för att lätt göra det själv skulle exempelvis möjliggöra att användaren lättare kan fokusera och uppmärksamma de som för stunden anses vara relevant. En annan tanke är att händelser som kan anses som kritiska skulle kunna uppmärksamas genom att använda faktiska ljud som associeras med händelsen. Att en krock har skett hade kunnat förmedlas genom det faktiska ljudet eller något som liknar.

E.6 Slutsatser

I detta avsnitt besvaras frågeställningarna i introduktionen utifrån resultatet och diskussionen.

Hur kan man gå tillväga för att designa ett gränssnitt som är mindre kognitivt ansträngande? Att designa ett mer fokuserat gränssnitt skulle kunna tolkas som en fråga om hur man designar ett gränssnitt som minimerar kognitiv ansträngning. Genom att konstruera gränssnitt som tar hänsyn till en användares kunskaper och har tanke bakom kan arbete utfört genom gränssnittet underlättas. Ansträngningen som krävs för att använda gränssnittet eller förstå informationen som syns kan minimeras genom att smart designa gränssnitten.

Vad finns det för tillvägagångssätt att påkalla uppmärksamhet hos en användare i ett gränssnitt? Att besvara denna fråga handlar också om att besvara vad för typ av uppmärksamning som en användare ska ta del av. Detta beror på kontexten, det kan vara enkelt som att en användare slår in fel lösenord sådan att textrutan skakar till för att påvisa att lösenordet var fel eller en varning om en kritisk situation. Animationer kan användas för att dirigera uppmärksamheten hos en användare genom att förmedla koncept eller förändringar genom

rörelse, detta tar då bort kravet för att beskriva i text vad som skett. Det går även att argumentera för att användaren genom detta också snabbare kan få förståelse för vad som skett genom detta gentemot om det förmedlades via text.

En faktor som alltid borde tas hänsyn till är att använda ljud för att förmedla vad som har hänt eller vad som sker. Som beskrivet i resultatet och diskussionen borde dessa ljud vara utformade på ett sätt som en användare intuitivt kan förstå i kontext.

Om detta skulle kopplas till Smart City kan uppmärksamhet påkallas genom att spela upp ljud för nya inhämtade händelser och visa animationer som gör att användaren snabbt kan plocka upp vad och vart något hänt.

F

Hur kan *Q-learning* användas för att testa ett användargränsnätsitt utvecklat i Unreal Engine 4? – Viktor Norgren

F.1 Introduktion

För detta projekt skulle gruppen implementera ett system där en användare kan se händelser i en stad. Några av de tänkta kunderna för systemet var samhällsorgan, till exempel räddningstjänsten. För att en kund ska vilja köpa produkten och vara nöjda med systemet måste den ha egenskaper som god pålitlighet. För att försäkra att systemet uppehåller de önskade egenskaperna ville gruppen därför testa det för att upptäcka problem som kunde orsaka oönskat beteende, till exempel krascher.

En av de viktigaste komponenterna av systemet är användargränssnittet, därmed ville gruppen testa det också. Det går självklart att testa användargränssnittet manuellt genom att testa all interaktion och verifiera innehållet. Detta tar dock väldigt lång tid och är opraktiskt, därfor ville gruppen automatisera testningen. En metod för att utföra automatisk testning av användargränssnitt är att använda en form av *Q-learning*, vilket är vad denna rapport kommer handla om.

F.1.1 Syfte

Syftet med denna rapport var att hitta en konkret metod för att testa användargränssnitt likt det gruppen använder i Smart City. Metoden skulle utföra automatisk testning och använda *Q-learning*.

F.1.2 Frågeställningar

1. Hur kan *Q-learning* användas för att utföra automatisk testning av ett användargränsnätsitt utvecklat i Unreal Engine 4?
2. När är det användbart att använda *Q-learning* för att testa ett användargränsnätsitt?

F.2 Teori

Det här kapitlet kommer innehålla information om de olika begreppen och teknikerna som resten av rapporten kommer ta upp.

F.2.1 Testning av Användargränssnitt

Målet med testning av användargränssnitt är att säkerställa att informationen som visas är korrekt och att försäkra att systemet beter sig korrekt när en användare interagerar med det [91].

F.2.2 Förstärkningsinlärning

Förstärkningsinlärning är en typ av maskininlärning, fördelen med den jämfört med andra tekniker är att den inte behöver ha någon fördefinierad data för att skapa en agent. Alltså behöver inte en utvecklare gå igenom all den data de vill använda och definiera vilket värde de har.

Den använder sig istället av *tillstånd*, *handlingar* och *belöning*. Principen med tekniken är att agenten ska utforska hur mycket belöning den får om den gör en viss handling i ett visst tillstånd, det kallas också för *tillstånd-handlings par*. Efter att ha utforskat "världen" kommer agenten ha kunskap om vilken handling den ska utföra när den hamnar i det tillståndet igen.

Agenten behöver något sätt att representera vilket tillstånd den befinner sig i. Den representationen bör innehålla all information som är relevant för att välja handling. Till exempel om agenten är en robot som ska ta sig från en punkt till en annan bör den veta var den befinner sig, vilken hastighet den har och i vilken riktning den pekar.

Robotens handlingar skulle kunna vara *ta ett steg med höger/vänsterfoten*, en annan kan vara *rotera 10° åt höger/vänster* och en skulle kunna vara *stå still*. Med hjälp av de tre handlingarna kan den förflytta sig i ett rum.

Agenten kommer ha en belöningsfunktion som räknar ut hur bra den presterar. I det tidigare exemplet är en funktion som ger mer belöning ju kortare avståndet är till slutdestinationen en passande belöningsfunktion. Den funktionen kommer leda till att agenten väljer handlingar som minskar avståndet till slutdestinationen, därmed förflyttar den sig dit [91, 92].

F.2.3 Q-learning

Q-learning är en form av förstärkningsinlärning. Den använder ett *Q-table* där den lagrar belöningen för alla *tillstånd-handlings* par. För att lagra alla värden på *tillstånd-handlings* paren behövs antalet handlingar multiplicerat med antalet tillståndsvärden. Detta innebär att *Q-learning* är opraktiskt när det finns många handlingar och tillstånd, eftersom det skulle kräva mycket minne att spara undan alla värden och ta lång tid att utforska världen.

Q-learning använder sig av en särskild matematisk formel vars delar förklaras nedan, där S_t är tillståndet och A_t är handlingen. Formeln tar i åtanke de möjliga framtida belöningarna som en handling kan ge [91, 92].

- $Q(S_t, A_t) < -$: Sätt värdet för *tillstånd-handlings* paret i agentens *Q-table* till värdet av allting till höger
- $Q(S_t, A_t)$: Det gamla värdet för *tillstånd-handlings* paret i agentens *Q-table*
- α : En konstant som heter *learning rate*
- R_{t+1} : Tillståndsvärdet som fås om agenten utför handlingen A_t
- γ : En konstant som heter *discount rate*
- $\max Q(S_{t+1}, a)$: Den högsta framtida förväntade belöningen om agenten utför handlingen i detta tillstånd

Hela formeln ser ut såhär: $Q(S_t, A_t) < -Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t)]$

Utforskning

För att fylla sitt *Q-table* utforskar agenten världen. När den utforskar väljer den handlingar slumpmässigt, den utgår alltså inte från sitt *Q-table*. Anledningen för detta är för att agenten ska, helst, testa alla *tillstånd-handlings* par. Ju fler par den har testat, desto mer vet den om världen. Detta leder till att den har större chans att välja rätt handling när utgår från sitt *Q-table*. Det är upp till utvecklaren att välja när agenten ska utforska och när den utgår från det den redan vet om världen [91, 92].

F.2.4 PySimpleGUI

PySimpleGUI är en modul till Python som underlättar skapandet av användargränssnitt [93].

F.2.5 PyAutoGUI

PyAutoGUI gör det möjligt för Python program att styra musen och tangentbordet, detta används för att utföra den automatiserade testningen [94].

F.2.6 OpenCV

OpenCV står för *Open Source Computer Vision Library*. Det är ett C++ API som tillåter en utvecklare att implementera program som kan hitta saker på skärmen. Det finns också en version av det som går att använda i Python [95].

F.3 Metod

För att hitta en lösning till frågeställningarna utfördes först en litteraturstudie med målet att hitta en metod som använder *Q-learning* för att utföra testningen.

F.3.1 Litteraturstudie

För att hitta relevanta artiklar användes sökmotorerna UniSearch och Google Scholar. De söktermer som användes var:

- GUI testing
- Autonomous GUI testing
- Automated GUI testing
- User interface testing

För att välja vilka artiklar att granska noggrannare användas sammanfattningarna för att se utifall de handlade om det önskade ämnet, en konkret metod som använde *Q-learning* för att testa användargränssnitt. Om sammanfattning av en artikel nämnde en metod lästes sedan slutsatserna för att se om metoden de tog upp faktiskt fungerade.

Detta pågick tills en metod var funnen där användargränssnittet som testades av metoden var snarlikt Smart Citys användargränssnitt.

F.3.2 Implementation

Efter litteraturstudien gjordes en implementation av den metod från litteraturstudien jag bedömde skulle testa Smart Citys användargränssnitt på bästa sätt. Denna bedömning gjordes genom att välja den metod där systemet som testades hade liknande interaktioner mellan användaren och systemet jämfört med de interaktioner Smart City har.

F.4 Resultat

Här presenteras den information som införskaffades genom litteraturstudien och implemen-tationen.

F.4.1 Litteraturstudie

Här presenteras den metod som togs fram genom litteraturstudien.

Föreslagna metoden

I artiklarna *Autonomous GUI Testing using Deep Reinforcement Learning* [92] och *Automating GUI Testing with Image-Based Deep Reinforcement Learning* [91] hittades ett exempel där forskare använder sig av *Q-learning* för att automatiskt utföra testning med målet att hitta buggar och orsaker till krascher genom användargränssnitt.

Med *Q-learning* kan en testningsagent implementeras med funktionaliteten att lagra vilka tillstånd den befinner sig i, vilka handlingar den väljer i de tillstånden och när problem uppstår. Med den informationen kan en utvecklare se vilka delar av systemet som troligtvis orsakar problemen, till exempel om utvecklaren ser att varje gång agenten trycker på en särskild knapp kraschar systemet, då är det nog värt att kolla noggrannare på den knappen.

Hur *Q-learning* används för att testa

Q-learning använder sig av tillstånd och handlingar. Tillståndet agenten befinner sig i definieras som de handlingar agenten har tillgång till just då. Handlingarna agenten har tillgång till är de möjliga interaktionerna den har med systemet. Det kan vara att den till exempel trycker på en knapp eller skriver in text. För att välja vilken handling den ska utföra använder agenten sitt *Q-table*, den väljer den handling som har högst belöning. Efter den har utfört en handling uppdaterar agenten det motsvarande belöning för det *tillstånd-handlings* den nyss utförde.

Agenten räknar ut *tillstånd-handlings* parets belöning med *Q-learnings* funktionen definierat i kapitel F.2.3 tillsammans med en särskild policy från *Autonomous GUI Testing using Deep Reinforcement Learning* [92], vilket är:

- Om $s = s'$: -1
- Om typen av handling ändras: $\frac{1}{x_a}$
- Annars: $\frac{1}{x_a} \cdot a_{s'}$

Alltså, om agentens nya tillstånd är detsamma som det förra tillståndet, blir belöningen: -1. Om typen av handling ändras, till exempel förra gången tryckte agenten på en knapp och den här gången skrev den text, då blir belöningen: ett delat på antalet gånger den handlingen tidigare har utförts. Annars blir belöningen: ett delat på antalet gånger handlingen som utfördes har utförts tidigare, multiplicerat med antalet nya handlingar som blev tillgängliga när agenten gick från det gamla tillståndet till det nya.

Denna policy leder till att agenten kommer prioritera de handlingar den har utfört lägst antal gånger i sitt nuvarande tillstånd och därmed styrs den till att utforska alla delar av systemet som den har tillgång till.

F.4.2 Implementation

Anledningen till att testningen implementerades i konventionell *Q-learning* istället för *djup Q-learning* var för att Smart City är ett relativt litet system, det finns endast ett lågt antal handlingar som agenten kommer kunna utföra. På grund av detta finns det också bara ett lågt antal tillstånd agenten kan befina sig i.

Initialisering av testsystemet

När programmet startas laddar det in bilder på knapparna utvecklaren vill att den ska kunna trycka på. När den laddat in bilderna skapar den en uppslagstabell där nycklarna är bildernas namn och värdet är en lista som innehåller en bool och ett heltal som börjar på noll. Testsystemet använder bildernas motsvarande bool för att hålla koll på om den har hittat den knappen och det motsvarande heltalet för att lagra hur många gånger systemet har tryckt på den knappen.

Handlingar

Handlingarna agenten har tillgång till är att trycka på knapparna den har hittat, den kan också välja att göra ingenting, vilket innebär att den väntar en kort stund. Om agenten inte hittar någon knapp alls utför den alltid handlingen "ingenting", eftersom det är det enda den kan göra. När agenten testar utför den handlingen som har den högsta belöningen i sitt *Q-table*, om alla möjliga handlingar har samma värde väljer den en slumpmässigt.

Tillstånd

Tillståndet agenten befinner sig i beror på de möjliga handlingarna den har tillgång till vid det tillfället, i detta fall är det de knappar den kan se och därmed trycka på. För att representera tillståndet används ett binärt tal. Ordningen som definierar vilken siffra som motsvarar vilken knapp bestäms när programmet laddar in bilderna och är den samma under hela testningen. I talet används en etta för att representerar att agenten har hittat den motsvarande knappen och en nolla representerar att den inte har hittat knappen.

Om systemet hade haft fem knappar och det hittade de två första knapparna men inte de tre sista skulle tillståndet se ut så här: 11000.

Belöning

Belöning räknas ut genom att använda *Q-learning* funktionen från kapitel F.2.3 och en modifierad policy från litteraturstudien. Skillnaden med policy:n är att den inte tar i åtanke om typen av handling ändras, eftersom agenten endast utför en typ av handling.

Testningen

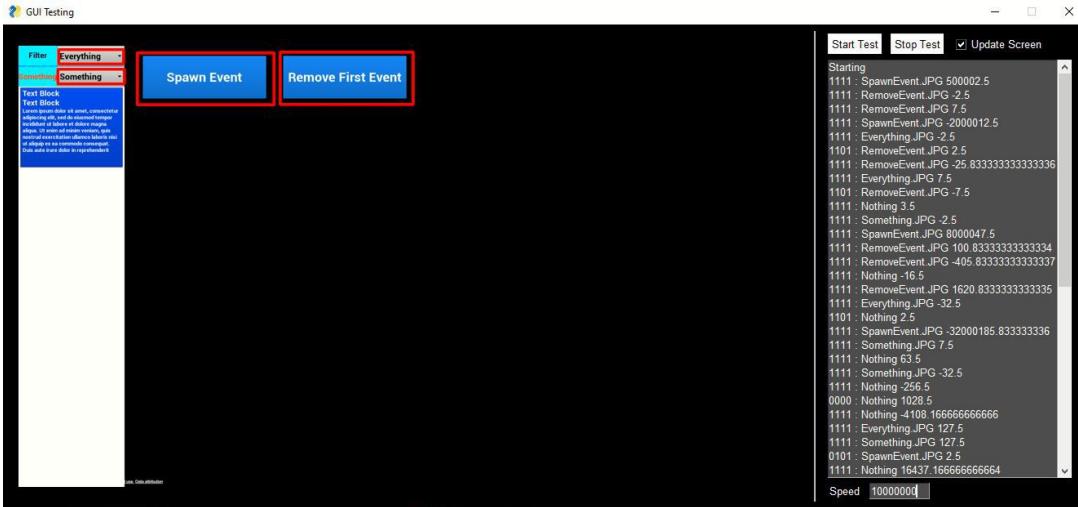
Sedan börjar själva testningen, det första agenten gör är att sätta alla bools i uppslagstabellen till falskt och tar en ny skärmdump. Sedan kollar den för varje knapp om den hittar den på skärmdumpen, den utför den kollen med modulen *OpenCV*. Om den är över 80% säker att den ser bilden anses den vara hittad och då uppdateras knappens motsvarande bool i uppslagstabellen till sant.

När den har letat efter alla knappar hämtas det nuvarande tillståndet med hjälp av uppslagstabellen. Sedan räknar agenten ut belöningen för det förra *tillstånd-handlings* paret och uppdaterar det motsvarande värdet i sitt *Q-table*. Anledningen till varför den uppdaterar den förras värde nu är för att belöningsfunktionen behöver veta vad det nya tillståndet är för att kunna räkna ut belöningen.

Agenten väljer sedan vilken handling den ska utföra genom att välja den handling som ger högst belöning enligt sitt *Q-table*. Efter agenten har valt en handling utför den det med hjälp av modulen *PyAutoGUI* och uppdaterar antalet gånger den utfört det i uppslagstabellen. Sedan börjar den om testnings processen, vilket pågår tills användaren av testsystemet stänger av det.

Användargränssnitt

För att underlätta felsökningen under implementationen och användandet av systemet skapades ett användargränssnitt som ses i Figur F.1. Det är implementerat i *PySimpleGUI*.



Figur F.1: Användargränssnittet för den egna implementationen när det används med en prototyp av Smart City:s användargränssnitt. På bilden finns det fyra knappar som agenten har hittat eftersom det är fyra knappar som har röda rektanglar runt sig

Den vänstra delen av användargränssnittet är en skärm dump av datorn som systemet körs på. De knapparna som testningen hittar på skärmen visas för användaren genom att det ritas en röd fyrkant runt knappen ovanpå skärm dumpen.

Till höger finns tre knappar på toppen, de är *Start Test*, *Stop Test* och *Update Screen*. *Start* och *Stop* används för att starta och avbryta testningen. *Update Screen* används för att uppdatera skärm dumpen som visas i användargränssnittet, att uppdatera skärm dumpen på användargränssnittet är inte ett krav för att testsystemet ska fungera. Under de är ett flöde, där finns en lista över alla tillstånd som agenten har befunnit sig i, vilken handling den valda att utföra i det tillståndet och den resulterade belöningen. I botten står det *Speed* och brevid det finns en ruta, där specificerar användaren av testsystemet hur snabbt testningen ska utföras, det värde som skrivs in motsvarar hur många millisekunder som testsystemet väntar efter den utfört en handling innan den utför nästa.

F.4.3 Testning av Smart City

Den enda delen av användargränssnittet som testningsagenten kunde testa var filterfunktionen eftersom gruppen fick slut på tid och hann därmed inte implementera de resterande funktionerna. Testningen av filtret visade att det kunde användas under en längre period utan att orsaka krascher, vilket var målet med testningen.

F.5 Diskussion

Här tas några funderingar och tankar upp om vad som hade kunnat göras annorlunda.

F.5.1 Brister i implementationen

En stor brist med det nuvarande systemet är att den enda handlingen den kan utföra är att trycka på knappar, vilket leder till att testningen kan vara begränsad. Ett exempel är en kommandotolk där användandet av detta testverktyg inte skulle kunna utföra någon, eftersom all inmatning en användare gör till systemet är via att de skriver text.

F.5.2 Q-learning eller Djup Q-learning

Skillnaden mellan konventionell *Q-learning* och *djup Q-learning* är att den senare använder sig av *neurala nätverk* för att lära sig om sin miljö, medan konventionell *Q-learning* använder en två-dimensionell tabell för att lagra allt den har lärt sig om världen.

Valet av konventionell *Q-learning* eller *djup Q-learning* för att testa användargränssnitt beror på storleken av systemet som ska testas. "För små (antal olika tillstånd), skulle konventionell *Q-learning* vara mer logisk istället för *Djup Q-learning*. Dock för större och mer komplexa system bådär det inte gott" [92].

Djup Q-learning är bättre för större system för att utvecklaren behöver inte bestämma antalet tillstånd i förväg eller skapa ett *Q-table*, vilket behövs göras i konventionell *Q-learning*. Utan ett *Q-table* minskar *Djup Q-learning* på minnesanvändningen då den inte behöver lagra ett stort antal värden. Det är också snabbare för det behöver inte söka igenom ett *Q-table* för att uppdatera och hämta värden [92].

F.5.3 Implementation i Python

Anledningen för varför testningen implementerades i Python var för att jag visste att alla verktyg som krävs för att implementera testsystemet fanns tillgängliga för Python.

Det som testsystemet behöver kunna göra är att hitta knappar på skärmen och trycka på knappar med musen för att simulera att en användare använder Smart City. Jag visste från tidigare erfarenhet att Python modulerna *OpenCV* [95] och *PyAutoGUI* [94] tillsammans hade denna funktionalitet, därmed valde jag att implementera i Python.

F.5.4 Mocking

Istället för att implementera ett externt testsystem som hittade knapparna på skärmen och använda en modul för att trycka på de hade jag kunnat använda *mocking* istället.

När *mock* testning implementeras används ett *mock objekt* istället för de riktiga mjukvaru objekten. Ett exempel som passar in i detta projekt hade varit att utvecklaren skapar ett objekt som simulerar användargränssnittet eller APIet. Om systemet har ett objekt som simulerar användargränssnittet hade systemet kunna ta emot påhittade inmatningar från det istället för ett riktigt användargränssnitt, vilket i denna testmetod hade varit att knappar blev tryckta [96].

F.6 Slutsatser

Här presenteras slutsatserna till frågeställningarna.

Hur kan *Q-learning* användas för att utföra automatisk testning av ett användargränssnitt utvecklat i Unreal Engine 4?

En metod för att automatiskt testa ett systems användargränssnitt med *Q-learning* är att använda de interaktioner en användare har med systemet som agentens handlingar. Med den tekniken kan en agent skapas som testar de möjliga interaktionerna en användare kan ha med ett användargränssnitt för att se om interaktionerna orsakar problem som till exempel krascher. Agenten har möjligheten att aktivt styras med hjälp av en belönings policy till att utforska de delar av systemet som är minst utforskade, därmed uppnås en hög testtäckning.

När är det användbart att använda *Q-learning* för att testa ett användargränssnitt?

Denna metod är användbar för att snabbt komma igång med testningen eftersom agenten behöver inte ha någon information om strukturen av systemet, den behöver endast ha information om vilka interaktioner med systemet den ska ha tillgång till. I implementationen från kapitel F.4.2 behövdes endast bilder på knapparna som skulle testas för att utföra testningen.



G

Analysmetoder för att lokalisera flaskhalsar i mindre mjukvarusystem likt Smart City - Gustav Peterberg

G.1 Introduktion

I denna rapport undersöks vilka metoder som existerar för att analysera mjukvarusystems prestanda som skulle kunna appliceras på mindre mjukvarusystem som Smart City. Smart City är vid rapportens skrivande ett relativt snabbt och responsivt system som mestadels begränsas av uppdateringsfrekvensen av informationskällorna. Det kan dock vara av intresse att analysera systemet för att finna andra potentiella flaskhalsar. I framtiden kan användare tänkas lägga till flera informationskällor till Smart City som då kommer hantera en mycket större mängd data. Potentiellt kan den ökade mängden data göra systemet långtsamt och användarovänligt. Det blir därför viktigt att strukturera systemet smart för att undvika onödiga flaskhalsar. För att lokalisera ett systems flaskhalsar måste en eller flera analyser utföras som går igenom systemets delar och lokaliseras dessa flaskhalsar.

G.1.1 Syfte

Denna rapport ämnar att hitta analysmetoder som lokalisera flaskhalsar och presentera dem lättförståeligt. Metoderna kan då tänkas implementeras av framtida utvecklare i till exempel Smart City men även liknande eller större mjukvarusystem.

G.1.2 Frågeställningar

1. Hur utförs en generell analys av flaskhalsar i mjukvarusystems prestanda i ett mindre mjukvaruprojekt?
2. Vilka analysmetoder för mjukvaruflaskhalsar finns för mindre och medelstora mjukvaruprojekt?

G.1.3 Avgränsningar

Utgångspunkten av analysen är att systemet ska använda sig av begränsad hårdvara. Ett system kan självklart ha väldigt låg latens och hög genomströmning om till exempel varje process hanteras av en egen CPU och så vidare. Dock blir detta väldigt dyrt och ohållbart om

välldigt många informationkällor läggs till. Därför begränsas analysen till att systemet körs på en dator som har prestanda i nivå med vad UE4 rekommenderar.

Eftersom Smart city vid rapportens skrivning inte är en färdig produkt kan systemet inte testas med de metoder och lösningar som resulterar av litteraturstudien. Därför siktat rapporten på att endast ta fram analysmetoder som är trovärdiga och applicerbara på systemet eller liknande större system.

G.2 Teori

Nedan förklaras den teori som behövs för att förstå vad flaskhalsar är och hur de kan uttryckas i mjukvarusystem.

G.2.1 Mjukvaruflaskhalsar

Flaskhalsar är ungefär definierade som något i ett system som begränsar dess prestanda. Detta kan häcka under specifika villkor och uttrycka sig i försämrad genomströmning i systemet och på så vis ett segare system för användaren. Flaskhalsar kan finnas av olika slag; långsam överföring av data mellan systemet till externa datorer, dålig synkronisering mellan funktioner inom systemet och dålig implementation av funktioner. Med synkronisering mellan funktioner menas dålig tajming av insignal och utsignal mellan funktioner. Till exempel är vissa funktioner inom ett system beroende av utsignalen från en annan funktion som i sin tur beror på en annan funktions utsignal. Detta skapar beroenden vilket resulterar i att många funktioner måste vänta på varandra. [97]

G.2.2 System

Ett mjukvarusystem tillhandahåller infrastrukturen som applikationen av mjukvaran kräver. Det är alltså den samling funktionalitet som uppfyller kraven applikationen satt. Smart City är systemet som uppfyller de krav som tagits fram från projektets marknadsanalys. [98]

G.2.3 Systemprestanda

Ett systems prestanda är främst uppmätt av användaren. Ett system kan ha många flaskhalsar men upplevas som responsivt och snabbt. En mer teknisk beskrivning är att prestanda i mjukvarusystem kan delas upp i fyra kategorier; tillgänglighet, latens, bandbredd, utnyttjande. Flaskhalsar kan alltså uttryckas sig i fyra former i mjukvarusystem. [99]

Tillgänglighet Tiden som systemet är tillgängligt för användaren, i termer av prestandatestning är detta när användaren inte alls effektivt kan använda systemet. Till exempel kan detta vara om användaren inte kan interagera med Smart City under tiden systemet uppdaterar alla händelser.

Latens Tiden det tar för en applikation att svara på användarens insignal. Detta innefattar hela den tid som systemet bearbetar insignalen till att användaren får ett svar. Till exempel kan detta vara den tid det tar för användaren att filtrera på händelser i Smart City.

Bandbredd Den takt som händelser i systemet kan utföras i, till exempel hur många händelser som kan läggas in i Smart City under en given tid.

Utnyttjande Teoretisk resurs användning, till exempel hur mycket används nätverket när Smart City uppdaterar alla händelser.

G.3 Metod

För att besvara frågeställningarna utfördes en litteraturstudie genom att söka information på internet.

G.3.1 Litteraturstudie

I studien hämtades information via publicerade artiklar på LiUs installation av arkivet *DiVA*. De söktermer som användes för att hitta information var följande.

- Performance testing methods
- Software bottlenecks analysis methods
- Finding software bottlenecks software systems

Källorna var relevanta för studien om de presenterade metoder för att hitta flaskhalsar i mjukvarusystem. För att ytterligare träffa alla metoder inkluderades metoder för att undersöka ett systems prestanda som är kopplat till flaskhalsar. Av de källor som resulterade i sökningarna valdes de med avseende på om de var peer-reviewed. Därefter granskades källans innehåll djupare, genom att skumma igenom källans sammanfattning och resultat. Om innehållet var relevant för att hitta flaskhalsar och upplevdes trovärdig togs källan med i rapporten.

En annan sak som noterade en kallas trovärdighet var antalet gånger källan tidigare varit citerad. I vissa fall användes även referenser ifrån källans referenser.

G.4 Resultat

I detta avsnitt presenteras de metoder som hittades av litteraturstudien.

G.4.1 Generell analys av systems prestanda för att hitta flaskhalsar

En generell arbetsgång för att analysera ett systems prestanda börjar enligt boken *Art of Application Performance Testing* [99] med att elicitera prestanda krav.

Prestandakrav

För att specificera hur systemet ska prestera måste prestandakrav sättas på systemet. Vissa systemfunktioner kanske är tillåtna att ta lång tid på sig medan andra måste hända relativt snabbt. För att formulera dessa krav måste någon form av användartester eller uppskattningar utföras på systemet vilket denna rapport inte kommer gå in på i detalj.

För att exemplifiera listas några påhittade krav som skulle kunna appliceras på Smart City nedan.

- Smart City ska kunna med några sekunders intervall uppdatera 500 händelser.
- Smart City ska kunna placera ut 1000 till 1500 händelser samtidigt i UE4.
- Smart City får ha högst 10 sekunder latens på serverförfrågningar.
- Filtrering av händelser får ta max en halv sekund.

Bygga tester

För att testa systemet kan olika tester implementeras, för det mesta autonoma för att få full test-täckning av systemet. En vanlig enkel metod för att analysera systemet under dessa tester är att mäta exekveringstider i systemet. Genom att mäta tiden från att en funktion får en insignal tills den returnerar en utsignal kan systemets prestanda mätas. Men även annan mätdata kan mätas som till exempel nätverksanvändning, latens för anrop över internet och genomströmning i form av antal operationer per sekund. För att testa systemets efter prestanda flaskhalsar finns fem olika tester listade nedan.

Bastest De första testerna som utförs på systemet för att ha datapunkter att mäta mot i andra tester. Detta kan vara att testa Smart City som den är vid rapportens skrivande med endast 5 datakällor, till exempel mäta hur snabbt systemet uppdaterar alla händelser.

Belastningstest Detta är ett klassiskt test som utmanar systemet med mängden data som genomströmmar. Till exempel skulle Smart City kunna testas genom att placera enligt tidigare krav 500 händelser i UE4 och mäta exekveringstiderna i systemet.

Blötläggningstest Ett test som tar lång tid att utföra. Genom att låta systemet köra under en viss belastning under en längre tid kan minnesläckor ackumulera eller okända flaskhalsar dyka upp vilket saktar ned systemet.

Isolationstest Tester som används för att hitta potentiella problem eller flaskhalsar. Genom att repetera specifika tester som tros ha en flaskhals kan de undersökas mer grundligt.

Dessa tester kan baseras på användarsituationer, till exempel utföra tester som om en användare hade använt systemet. Ett exempel för Smart City hade kanske varit att användaren hade 500 informationskällor som skulle placeras och uppdateras kontinuerligt. Men även felaktiga insignalen till systemet kan vara värd att testa ifall systemet presterar dåligt vid kontroller av giltiga värden.

Kan vara värt att nämna är att låta insignalerna till systemet genereras på en separat dator. Detta för att låta systemet köras som tänkt utan att andra program tar upp hårdvaruresurser. Till exempel ska datakällorna för Smart City genereras på andra externa datorer vid tester.

Analysera resultatet

Efter att tester utförts och resultatet sparats kan informationen börja analyseras efter flaskhalsar.

Genom att rita ut grafer och tabeller över exekveringstiderna för vissa funktioner eller kluster av funktioner kan en översikt fås. Exekveringstiderna kan då jämföras mot bas-testerna för att observera avvikelser där systemet kanske presterar extra dåligt. Dessa funktioner eller vid dessa kluster av funktioner finns sannolikt en flaskhals i systemet.

För att analysera nätverksanvändningen kan överföringshastigheten och latensen för interne-tanrop ritas upp på grafer. Även här kan avvikelser observeras för att möjliga hitta flaskhalsar där systemet presterar dåligt. Det kan även vara värt att beräkna medelvärdet för latens och nätverksanvändning för att få en översikt och bestämma om de är för långa respektive hög.

G.4.2 FOREPOST

FOREPOST - Feedback ORiEnted PerfOrmance Software Testing är en metod framtaget för att undersöka mjukvarusystems flaskhalsar [100]. Ett mjukvarusystem kan undersökas med FOREPOST genom att analysera funktionsstrukturen i systemet. Metoden utförs i 5 steg varav

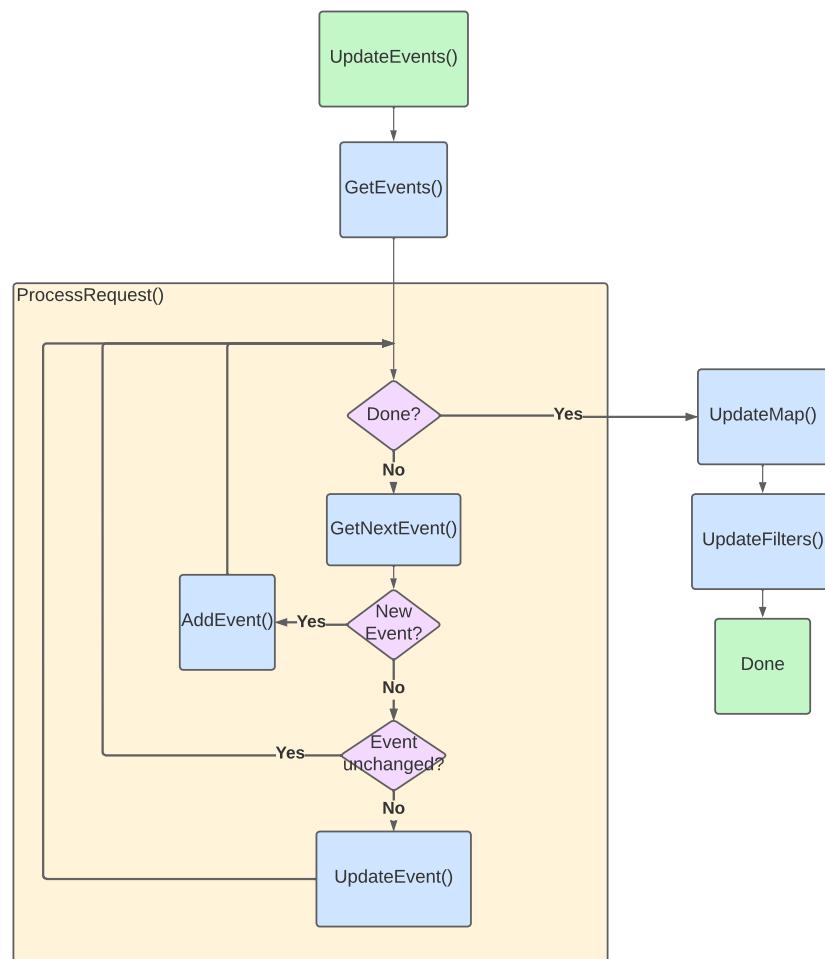
vissa repeteras ett antal gånger. Det börjar med att undersöka operationsvägar i systemet genom att testa med en mängd slumpmässiga och förbestämda insignal till systemet likt i G.4.1. För varje insignal till systemet skapas en slags ”profil” som antecknar dess väg genom systemet och hur länge det tog att exekvera. Ur denna analys fås en rad fakta om funktioners exekveringstid, genomströmning och antal gånger en metod blivit kallad. Genom att analysera dessa profiler kan de delas upp i bra och dåliga med hjälp av uppsatta regler. Dessa regler kan till exempel vara kontroller om vissa metoder som kräver mycket beräkningskraft har exekverat. En bra väg i det här fallet räknas som beräkningsmässigt tung och dålig väg som lätt.

Dessa regler sätts upp genom maskininlärning som lär sig vilka funktioner eller vägar som är beräkningsmässigt tunga eller lätta. Maskininlärningen har även ett mål att söka de mest beräkningstunga vägarna och kan då skippa de dåliga vägarna. Inlärningen blir bättre desto mer indata som kan testas vilket dock betyder att väldigt många operationsprofiler måste skapas vilket tar mycket tid. När dessa regler och profiler satts upp kan flaskhalsar börja urskiljas. Dock förekommer flera metoder och funktioner i olika profiler vilket kan göra det svårt att urskilja vilka som är de mest signifikanta flaskhalsarna. För att lösa problemet som är av typen *Blind Source Separation* (BSS) används *Independent Component Analysis* (ICA). Algoritmen sorterar ut de potentiella flaskhalsarna ur profilernas resultat.

Det finns alltså flera problem som måste lösas för att hitta flaskhalsar; sätta upp operationsprofiler och konstruera regler för att dela upp de i bra och dåliga, urskilja flaskhalsarna ur dessa profiler i kombination med reglerna.

Skapa profiler

För att skapa exekveringsprofiler måste alla eller specifika insignalers operationsvägar samlas in. Insignalerna kan vara slumpmässiga men även reflektera en användares insignal till systemet. Detta kan göras med redan existerande verktyg som ofta kostar pengar. Eller konstruera ett verktyg som mäter alla exekveringstider för såväl metoder som större funktioner i systemet. Samla även in antal gånger som metoder och funktioner anropas och antalet trådar använda samtidigt. För att demonstrera metoden kommer ett exempel på en hypotetisk funktion hos Smart City med dess operationsväg.



Figur G.1: Exempel på en operationsväg genom ett system.

Genom att låta systemet uppdatera ett varierat antal Händelser i UE4 kontinuerligt enligt figur G.1 kan dess prestanda under last mätas. Nedan följer ett påhittat resultat av en mätning på denna operationsväg under last.

Tabell G.1: *Påhittade potentiella resultat efter mätningar av exekveringstider på operationsvägar genom figur G.1*

Funktion	Antal händelser	Medelvärde av antal anrop	Medelvärde exekverings-tid (s)	Längsta exekveringstid
ProcessRequest	50	1	1.1	1.8
ProcessRequest	150	1	2.6	3.6
ProcessRequest	400	1	7.4	10.1
AddEvent	50	30	0.005	0.1
AddEvent	150	91	0.02	0.07
AddEvent	400	370	0.09	0.12
UpdateEvent	50	22	0.01	0.08
UpdateEvent	150	76	0.07	0.16
UpdateMap	50	1	0.9	0.08
UpdateMap	150	1	1.9	2.2
UpdateMap	400	1	3.1	4.5

Uppdelning av exekveringsfall

När alla profiler skapats ska de olika profilernas operationsvägar delas upp för att identifiera vilka som potentiellt kan innehålla flaskhalsar. Profilerna delas upp i dåliga och bra testfall där ett dåligt testfall är då funktionerna exekverat snabbare och ett bra testfall då de exekverat längsammare. Detta görs genom att gruppera upp funktioner utefter medelvärdet av deras exekveringstid. Resultatet blir på formen $V_{I_1}, \dots, V_{I_k} \rightarrow T$ där V_{I_m} är värdet från insignalen I_m och $T \in [G, B]$ där G och B representerar bra och dåliga testfall.

För att ge ett exempel kommer några påhittade resultat listas nedan. Funktionerna figur G.1 tilldelas ett I_k och dess värde V_{I_k} . Utifrån detta följer dessa operationsvägar med dess resulterande T.

Tabell G.2: *Uppdelning och namngivna funktioner med dess tillhörande värde från resultatet i tabell G.4.2*

Funktion	funktion I_k	Antal gånger anropad V_{I_k}
ProcessRequest	I_1	1
AddEvent	I_2	150
UpdateEvent	I_3	58
UpdateMap	I_4	1

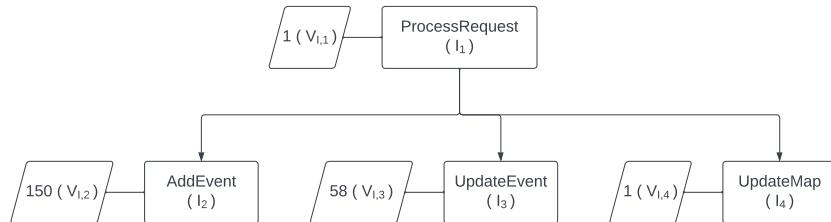
Utifrån dessa funktioner och värden kan påhittade operationsvägar formas som till exempel $V_{I_1}, V_{I_2}, V_{I_3}, V_{I_4} \rightarrow G$. Detta utförs egentligen helst på ett långt många olika operationsvägar och funktioner.

Erhålla regler

För att utvinna profilregler i FOREPOST används maskinlärningsalgoritmen RIPPER - Repeated Incremental Pruning to Produce Error Reduction [101]. Algoritmen bygger på en tidigare algoritm Incremental Reduced Error Pruning (IREP) men kräver inte lika mycket beräkningskraft. Algoritmens insignaler är resultatet av de exekveringsvägar som skapades sedan tidigare, $V_{I_1}, \dots, V_{I_k} \rightarrow T$. Algoritmens resultat blir regler på formen $I_1 \odot V_{I_1} \bullet I_2 \odot V_{I_2} \bullet \dots \bullet I_2 \odot V_{I_k} \rightarrow T$, där \odot symbolisera en av relationsoperatörerna, (till exempel $<$ och $=$) och \bullet symbo-

liserar logiska operatörer (till exempel \vee och \wedge). Reglerna utgörs av villkor som till exempel att om `UpdateEvent` anropas fler än 40 gånger leder det till ett bra testfall.

Insignalen kan ses som ett operationsträd med noder som är funktionerna. Nodernars värde blir antal gånger funktionen blir anropad i exekveringen, se figur G.2 nedan



Figur G.2: Exempel på operationsträd där noderna är funktionen och har värdet av antal gånger de anropats i den exekveringen.

Ripper itererar genom trädet och följer en slags implementation av söndra och härska tekniken för att dela upp problemet. Metoden för att använda RIPPER beskrivs delvis inte men en generell arbetsgång kan beskrivas. Algoritmen börjar med fem listor, en tom *regellista*, två *växande listor* en för bra testfall respektive dåliga och två *beskärningslistor*, en för bra en för dåliga testfall. Villkor för regler läggs till i regellistan girigt genom att iterera genom trädet och testa att öka eller minska nodernas värde. Villkor på formen ($I_1 > V_{I_1}$) läggs till utifrån nodens värde. Villkoren välj genom att lägga till de som maximerar informationsfångsten, alltså inga dubletter av villkor förekommer. De villkor vars nodvärde ökar läggs till i växande listan och de som minskar läggs till i beskärningslistan.

När en regel skapats kommer den direkt bli beskuren för att ta bort redundans enligt följande definition. $v(\text{regel}, \text{beskärningslistaBra}, \text{beskärningslistaDålig}) = \frac{p-n}{p+n}$ där p är antalet villkor i BeskärningslistaBra som täcks av regeln och n antal villkor i BeskärningslistaDålig. Detta pågår tills värdet på v inte förbättras längre. Skapa sedan regler tills det inte finns bra tester kvar, om nuvarande regel innehåller 64 mer villkor är den regeln med minst antal villkor eller om felfrekvensen är större än 50% vilket inte förklaras hur det beräknas.

När flera regler skapats beskärs regler en gång till för att ta bort redundans mellan reglerna.

G.4.3 Urskilja flaskhalsar

För att urskilja vilka delar av systemet som är flaskhalsar analyseras de regler som satts upp. Om en funktion anropas många gånger i de bra testfallen och anropas inte alls eller väldigt lite i de dåliga testfallen kan den misstänkas vara en flaskhals. Denna information är inbakad i alla regler vilket gör det svårt att urskilja vilka funktioner som påverkar systemet mest. Detta lösas genom att utföra en ICA - Independent Component Analysis som identifierar flaskhalsarna ur reglerna. Detta görs genom att lägga normaliserade vikter på villkoren med avseende på tre kriterier nedan.

- Hur många gånger funktionen anropas.
- Hur lång tid funktionen exekverar minus den summerade exekveringstiden av alla funktioner som anropas av funktionen.
- Antal funktioner som anropas från funktionen.

Därmed väger vissa regler mer än andra att exekvera och flaskhalsar kan enklare urskiljas. För att använda dessa vikter används (ICA). Funktioner och operationsvägar bestäms av högre abstraktions behov, till exempel "logga in" eller "uppdatera händelser" vilket resulterar i att vissa operatonsvägar överlappar med en högre abstraktionsnivå som är användarens behov. Med detta i åtanke kan de mest viktiga systemapplikationerna identifieras. Problemet blir då att separera ut regler som är länkade med en viss systemapplikation. Detta problem kallas Blind Source Separation (BSS) och är implementerad med ICA.

ICA

I metoden används en ICA matris, där processen beskrivs av ekvationen $x = A \times s$ där x är matrisen som innehåller villkoren från reglerna, A är operationsmatrisen som kan tänkas sprida ut dessa villkor på flera regler och s är de potentiella flaskhalsarna vi vill åt. Elementen i matrisen x ges av $x_i^j = \sum_{k=1}^n \lambda_k \times M_{i,k}^j$ där λ är de normaliserade koefficienterna som är beräknad för hela matrisen x och som säkerhetsställer att $0 \leq x_i^j \leq 1$. M är de värden som funktionens vikt tar hänsyn till, för Smart City är till exempel $M_{i,1}^j$ är antalet gånger funktionen j blivit anropad i operationsvägen i . $M_{i,2}^j$ är totala exekveringstiden för funktionen j minus exekveringstiderna för alla funktioner som anropas i funktionen j . $M_{i,3}^j$ är antalet funktioner som anropas ur funktion j .

A matrisen kan liknas vid de operationsvägar som finns för funktionerna i systemet. Elementen i A matrisen, A_p^q är vikten som varje profil p bidrar med i exekverande kod som implementerar funktionen q . Matrisen förklaras inte med något exempel eller konkret metod att ta fram.

Genom att lösa ut matrisen s kan flaskhalsar identifieras ur de regler som skapats sedan tidigare. Detta är sista steget av analysen.

G.4.4 Manuella tester

För att testa ett system efter prestanda brister och flaskhalsar kan manuell testning löna sig för små system [99]. Men manuell testning kostar mycket resurser och är tidsödande. Det rekommenderas därför att använda sig av automatiserad testning för de allra flesta projekt. Större mjukvaruprojekt måste använda automatiserad testning där en dator utför testerna. Detta för att få någon form av täckning av systemet och utföra det inom rimlig tid.

G.5 Diskussion

I detta avsnitt diskuteras resultatet av de metoder och processer som ska besvara frågeställningen.

G.5.1 Litteraturstudien

För att hitta information hämtades källor via de sökmotorer och söksträngar som nämns ovan i avsnitt G.3. Jag tror urvalet av källor och dess dokument har varit okej utvalda med avseende på trovärdighet och relevans därför att urvalsriterierna var rimliga och kunde uppfyllas. FOREPOST visade sig i djupare analys sakna bitar av information som visade sig viktig för att förklara dess funktion. I efterhand skulle kanske en till metod lagts till för att verkligen ge exempel på lösningsmetod. En analys av mängden uppptagna hårdvaruresurser skulle kunna utöka rapporten och vara relevant för Smart City.

Vissa söksträngar gav flera resultat och med hänsyn till tid valdes några enstaka att undersöka. Vid sökning av analysmetod för flaskhalsar valdes endast en generell analys för prestanda och FOREPOST då metoderna var lika andra metoder och gav ungefär samma resultat. Jag

bedömde det räckte med dessa två för att täcka ändamålet. I annat utförande skulle FOREPOST kunna jämföras med andra automatiserade metoder för att undersöka dess effektivitet med avseende på typen av system.

G.5.2 Resultat

Ett generellt sätt att hitta prestandaflaskhalsar i mjukvarusystemet hittades enligt resultat. Den beskriver översiktligt en arbetsgång för att hitta flaskhalsar men verkar inte ha samma analytiska bredd som användningen av till exempel FOREPOST. Men generella analysen kanske räcker för att undersöka mindre mjukvaruprojekt då denna metod är mycket enklare att implementera och förstå. Att implementera FOREPOST i nuvarande Smart City skulle säkert fungera bra och hitta några flaskhalsar. Dock kan det tyckas vara en överflödig analys för hur stort Smart City faktiskt är. I artikeln om FOREPOST nämner de tester på mjukvarusystem med flera tusen funktioner varpå Smart City har en handfull. För Smart City kanske den mer simpla generella metoden räcker för att hitta relevanta flaskhalsar som påverkar systemet mest.

G.5.3 Autonoma tester eller manuella tester

Autonoma metoder tillåter analys av väldigt många fler insignaler till systemet än en mänskliga skulle kunna göra inom en rimlig tid. En mycket större mängd insignaler testade betyder fler sökta vägar i systemet vilket kan potentiellt leda till fler funna flaskhalsar. Dock kan väldigt många insignaler anses vara redundanta eller likvärdiga vilket betyder att väldigt många dåliga insignaler och rutter undersöks. En erfaren mjukvarutestare kan mycket mer effektivt undersöka möjliga insignaler som skulle kunna visa flaskhalsar i systemet. Under litteraturstudien hittades mycket lite information om användningen av manuella flaskhalsanalyser. Att manuellt analysera är ansett svårt och vara en kostsam analys beräkningsmässigt att utföra. Automatiska tester verkar vara lite jobb att sätta upp för första gången. Men efter en initialt större arbetsinsats i jämförelse med manuella analyser kommer autonoma tester löna sig.

G.5.4 Nutida och framtida effekter av åtgärdade flaskhalsar i Smart City

Jag tror effekten av att implementera FOREPOST eller den generella analysen på Smart City skulle definitivt mildra effekten av flaskhalsar i systemet men självklart inte eliminera de. Effekten av dessa verktyg verkar vara större desto större systemet och mängden data som behandlas är. Vid rapportens skrivande är Smart City inte ett väldigt stort system, vilket medför att i dagsläget hade effekten förmodligen inte varit väldigt stor. För ett system likt Smart City i framtiden som är mycket större och behandlar mycket mer data kan dock detta bli relevant. Om FOREPOST skulle identifiera några flaskhalsar som sedan åtgärdas i ett sådant system skulle produkten bli mycket bättre. En förbättring av systemets beräkningstid med till exempel 12% skulle vara betydligt märkbar om uppdateringstiden för alla händelser sänktes från 300 sekunder till 264. En annan fördel är att förbättringen inte kostat något i hårdvara. Systemet kan dock fortfarande begränsas om till exempel UE4 är prestandamässigt snabbare än nätverkets överföringshastigheter efter applicering av FOREPOST. Då finns det fortfarande en flaskhals kvar eftersom UE4 eller nätverket kommer vänta på den andre. Dock borde generella prestandan i systemet fått en positiv nettoeffekt.

G.5.5 Förbättringsmöjligheter

Förutom att analysera systemets struktur skulle en ny modul kunna läggas till explicit för schemaläggning av processer. Detta kan innefatta datakällors uppdateringsfrekvens, då alla datakällor inte behöver ha maximal uppdateringsfrekvens hela tiden. En avvägning skulle

kunna användas där modulen schemalägger optimala anrop av information. Annars skulle en utvecklingsmöjlighet vara att designa hårdvara för systemets ändamål. Delsystemen i systemet skulle kunna distribueras ut på olika hårdvukomponenter och därmed fokusera på enskilda uppgifter. Potentiellt skulle detta öka generella beräkningskraften och då göra systemet snabbare. Hämtning av information från informationskällor kan fördelas på flera datorer, databasen på en annan och UE4 visas på en. Dock kan detta potentiellt också öka systemets dataöverföringstid då information måste skickas mellan dessa komponenter över större sträckor än om alla komponenter varit inom samma dator. Det tar då beräkningskraft att packetera data, skicka och ta emot när data istället skulle kunna tas emot via en databuss. Området skulle potentiellt vara ett rimligt förbättringsområde.

G.6 Slutsatser

Här besvaras frågeställningarnas resultat som erhölls från litteraturstudien.

G.6.1 Hur utförs en generell analys av flaskhalsar i mjukvarusystems prestanda i ett mindre mjukvaruprojekt?

Under litteraturstudien hittades en metod för generell analys av ett mjukvarusystems prestanda och därmed flaskhalsar. Genom att mäta exekveringstid för ett systems funktioner under last kan dess prestanda mätas. Lasterna kan konstrueras utifrån en hypotetisk användare som ger insignaler till systemet eller slumpmässiga insignaler. Utifrån den resulterande datan kan en statistisk analys utföras för att lokalisera var och när prestandan i systemet blir sämre. Detta med avseende på avvikelse från medelvärdet av funktioners exekveringstid. På så sätt kan flaskhalsar i system lokaliseras på ett relativt enkelt sätt men kanske inte analyserar systemet grundligt.

G.6.2 Vilka analysmetoder för mjukvaruflaskhalsar finns för mindre och medelstora mjukvaruprojekt?

Det finns automatiserade metoder för att analysera och lokalisera ett systems flaskhalsar. Under studien hittades FOREPOST, en automatiskt metod som kan analysera stora system relativt snabbt och hitta flaskhalsar som kan vara annars svåra att lokalisera. Metoden går igenom systemets operationsvägar och drar slutsatser om var systemets prestanda begränsas. Eftersom metoden är automatisk kan de allra flesta insignaler och operationsvägar i systemet analyseras vilket resulterar i mycket större andel test av systemet än om den undersöks manuellt av en testare. På så sätt kan annars svårfunna flaskhalsar identifieras.

H

Versionshantering av Unreal Engine 4 projekt med Git-Anna Zhao

H.1 Introduktion

Vid utveckling av ett system, särskilt i grupp, är versionshantering och möjligheten att lätt kunna dela och ta del av varandras arbete en viktig aspekt. Detta kan låta lättare än vad det är, särskilt när man pratar om system utvecklade i spelmotorer. Dessa projekt har oftast utmaningen att bland annat hantera binärfiler och större mängder data i form av till exempel grafik och 3D objekt. Denna rapport kommer att undersöka dessa utmaningar och titta närmare på hur verktyget Git sköter versionshanteringen av system utvecklat i Unreal Engine 4 (UE4).

Tillkännagivande

Jag vill tacka Gustav Peterberg som har korrekturläst min rapport.

H.2 Syfte

Syftet med denna rapport är att undersöka de unika aspekterna vid versionhantering av system utvecklat i spelmotorer som UE4. Rapporten ska även undersöka hur versionshanteringsverktyget Git och GitLab hanterar dessa unika aspekter.

H.3 Frågeställning

1. Vilka utmaningar finns vid versionshantering av system utvecklat med spelmotorer?
2. Hur hanterar Git versionshanteringen av system utvecklat med Unreal Engine 4?

H.4 Teori

Det här avsnittet kommer ta upp begrepp, termer och koncept som läsaren behöver för att förstå resterande rapport.

Plain text

I denna rapport refereras *plain text* till en sekvens av tecken som är läsbar för människan. Det finns få undantag i form av till exempel karaktärer som indikerar *tab* eller en ny rad. En *Plain text* fil ska kunna öppnas, läsas och ändras av en standard textredigerare.

Binärfiler

Det som definierar en binär fil är att de innehåller någon sekvens av bitar som inte nödvändigtvis representerar *plain text* se figur H.1. Många system som kodar binärt är även unika och fungerar för specifik hård- eller mjukvara. Eftersom den binära delen av filerna inte är läsbar av människan är det svårt att direkt korrigera eller göra ändringar i dem. [102]

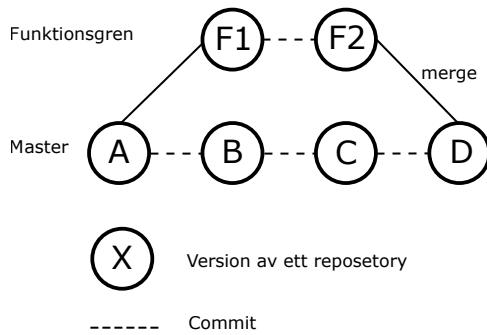
Figur H.1: Binärfil öppnad i en textredigerare.

3D scenbeskrivning

I en 3D scenbeskrivning ingår inte bara geometrin av objekt utan den refererar även till entiteter som en kamera och ljus [103].

Git merge

Målet vid en *git merge* är att sammantagna två versioner av ett *repository*. Detta kan både ske när utvecklaren vill sammantagna två olika utvecklars kod eller olika grenar. Vid en sammantagning integreras utvecklaren en av versionerna in i den andra. En arbetsgång som kan leda till en *merge* kan ses i figur H.2. Under sammantagningen kommer ändringar som inte överlappar, till exempel ändringar i området av filen som endast ena versionen har, skrivas i den slutliga versionen verbatim. Detta betyder att om filens struktur har en stor betydelse kommer inte sammantagningen ta hänsyn till det. I fall där båda versionerna har ändringar i samma område av en fil och det skapar en konflikt, har Git själv inte förmågan att välja den ena versionen över den andra. För att lösa detta kommer en utvecklare behöva besluta om vilka ändringar som ska behållas eller tas bort. Förutom att gå in i filen för att lösa konflikten kan användaren även direkt välja den ena versionen över den andra vilket är användbart i fall av binärfiler. [104, 105]



Figur H.2: Ett exempel av en arbetsgång där merge används.

Git bash och Git GUI

Git bash och Git GUI är verktyg för att utföra Git kommandon. Git bash är en kommandotolk som liknar standard kommandotolken i LINUX och UNIX miljöer. Git GUI är ett grafiskt användargränssnitt som kan integrera med Git.

H.5 Metod

Denna rapport kommer att baseras på en litteratursökning som kommer att kompletteras med egen erfarenhet som samlats under projektets gång. Ett enkelt fristående experiment kommer även utföras för att samla information om systemet som skapats i UE4.

Litteratursökning

En generell litteratursökning utfördes kring ämnena i listan nedan. Främst försöktes litteratur hittas i databaser som DIVA, Linköping University Library, Google scholar och IEEE Xplore. De första fem till tio sidorna av sökresultaten undersöktes. Vid litet utbud av relevant litteratur som behandlar ämnet användes sökmotorn Google eller en officiell sida angående ämnet. Under dessa sökningar kommer det ske ett extra noga urval av litteratur, metoden för detta beskrivs i nästa stycke.

- Versionshantering i spelmotorer
- Binärfiler
- versionshantering av binärfiler
- Git
- UE4

Urval av litteratur

Vid litteratursökningen skedde det en selektion av literturen baserat på kriterierna nedan:

- Äkthet
Med äkthet tar man hänsyn till vem som ligger bakom källan, varför källan skapades och hur informationen kom till [106].
- Tid
Tidskriteriet i denna rapport handlar om när informationen är skapad men eftersom mycket är faktat kollas även hur sannolikt det är att informationen har ändrats.

- **Beroende**
Beroende i denna rapport kommer undersöka hur många gånger informationen har bearbetats, alltså om den är en förstahandskälla eller tagen från andra källor.
- **Tendens**
Tendens tar hänsyn till om upphovsmannens intresse kan påverka informationen på ett vinklat sätt [106].

Förutom dessa kriterier kommer även ett urval ske efter en granskning av sammanfattningsnär det finns eller en snabb översikt av innehållet för att besluta om källans relevans.

Insamling av egna erfarenheter

Under projektet användes Git för att versionshantera gruppens system som utvecklades i UE4. Detta gav möjligheten att observera och undersöka hur det fungerade i praktiken. Intressanta erfarenheter dokumenterades under projektets gång i ett dokument. Under projektets gång undersöks även de filer som producerades för att se vilka egenskaper de hade.

Fristående experiment

Ett enkelt fristående experiment från projektet utfördes även i UE4 för att undersöka skillnaden mellan de filer som skapades när jag använde *Blueprints* och C++. I detta experiment kollade jag på filformat, storlekar på filerna och hur de hanterades av UE4 samt Git.

H.6 Resultat

I detta kapitel presenteras resultatet av litteratursökningen, sammanställningen av personliga erfarenheter och det fristående experimentet.

H.6.1 Versionshantering för spelmotorer

Genom projektets gång har gruppen uppmärksammat att system utvecklade i spelmotorer skiljer sig från många andra mjukvaruprojekt. De hanterar inte bara kod i form av text utan även binärfiler som kan beskriva till exempel polygonytor (*Polygon mesh*), texturer och 3D scenbeskrivningar. Hantering av binärfiler medför vissa utmaningar, som till exempel när binärfiler arbetas på parallellt kommer det leda till svåra *merge* konflikter. Dessa konflikter har inte samma lösningsalternativ som textbaserad kod. De binära filer som genererades i detta projekt visades även vara mycket större än de C++ filerna som skapades i projektet.

3D modeller

I de flesta fallen formateras 3D modeller till binärfiler. I vissa fall om en *3D scene* är uppdelad i flera filer måste även relationera mellan dessa filer sparas [107].

UE4

Blueprint är UE4:as visuella programmeringsspråk där den traditionella koden representeras i form av noder som kopplas med varandra. Mycket av det utvecklaren kan skapa genom C++ kod kan även skapas med *Blueprints*, allt från logik till visuella designer. Från egen erfarenhet var det betydligt enklare att jobba med *Blueprints* än med ren C++ programmering i den skala projektet hade.

När filerna som genereras från UE4 inspekterades kunde jag observera att *Blueprints* samt material-filer sparas som binärfiler med ändelsen *.UASSET*. Filerna som beskriver vad som finns i världen bestod också av en binär fil av typ *.UASSET* och en annan fil som av typen *.UMAP*. *.UMAP* tillhör i filkategorin spelfiler och har ingen designeras filformatering [108]. Vid en närmare inspektion av *.UMAP* var strukturen av innehållet mycket lik *.UASSET* i aspekten att innehållet inte var ren *plain text*.

Stora filer

Vid inspektion av projektfilerna kunde jag se att binärfiler från bland annat *Blueprints* och material kan ta upp från 25 KB till 1298 KB. Detta är betydligt mer än vad C++ filerna tar då deras minne ligger under 1 KB där den största filen vi skapade i projektet ligger runt 9 KB. En sak att ha i åtanke är att det skapas två C++ filer för ett objekt, en *C Header Source File* och en *C++ Source File*.

Vid ett mer kontrollerat experiment där två likvärdiga objekt skapades med både C++ och *Blueprints* kunde jag se att *Blueprint* hade storleken 19,5 KB och C++ filerna tillsammans endast hade 1,077 KB. De två objekten som skapades var start filer som ärvde UE4:as *Actor class*, där ingen extra kod laddes till av mig. I detta experiment skapades även en material-fil som hade storleken 90,1 KB utan någon extra tillägg av mig.

I detta projekt använde projektgruppen mycket material från det externa verktyget Cesium. Detta gjorde att stora delar av projektet inte behövde versionshanteras eftersom alla i gruppen använde till exempel samma glob och karta. En lokal version av Cesium ligger runt 7,65 MB, utav detta ligger material på ca 3,04 MB och texturer runt ca 1,91 MB [109].

H.6.2 Git som versionshanteringsverktyg

När ett system versionshanteras med Git skapas ett lokalt *repository*. Genom att använda exempelvis GitLab kan utvecklaren spara detta på en server vilket gör det möjligt att spara ens arbete externt. Genom detta kan projektet även delas med flera användare. [110] När en utvecklare klonar ett *repository* från en server kommer hela historiken av detta *repository* hämtas ner på utvecklarens lokala enhet [105].

GitLabs gratis *repository* storlek gräns ligger på 5 GB [111].

Git LFS

Med *Git LFS* ersätts stora filer av en textfil, som innehåller en pekare, i projektets *repository* medan den verkliga filen sparas på en separat fjärrserver. Detta resulterar i att vid versionhantering sparas inte de verkliga filerna i utvecklarens *repository* och därmed kommer de även inte sparas i historiken. På grund av detta kommer utvecklarens lokala version av ett *repository* inte spara versioner av binärfiler som inte används för stunden. [18, 112]

Från de observationer jag har utfört kom jag fram till att även om binärfiler sparas som textfiler i utvecklarens *repository* sparas hela binärfilen av den gamla versionen på fjärrservern när en ändring sker. I GitLab sparas dessa filer på GitLabs egna servrar [113]. Från detta togs slutsatsen att minnet utvecklaren använder vid *Git LFS* räknas in i deras *repository* storleksgräns.

Låsa filer

GitLab och *Git LFS* erbjuder funktionen att låsa filer. När en fil är låst av en utvecklare förhindrar det andra utvecklare att modifiera filen genom att till exempel *pusha* upp ändrad kod till den filen, genom att försöka *merga* in en annan version av den filen eller på något annat sätt. GitLab tillåter utvecklaren att låsa filer och mappar i originalgrenen om utvecklaren har ett *GitLab Premium* konto eller högre. Originalgrenen är oftast den gren som skapas när系统的版本管理被初始化。*Git LFS* låter utvecklaren skapa exklusiva lås som fungerar över alla grenar. Detta görs med hjälp av filen *.gitattributes*, i denna fil talar utvecklaren om vilka filtyper som är låsbara. Detta måste *pushas* upp till den externa *repository* för att ändringarna ska kunna ta effekt. När filtyperna är registrerade som låsbara kommer dessa filer vara skrivskyddade som standard och för att kunna ändra de måste utvecklaren låsa filen. [114].

Unreal

Efter användning av UE4 har jag observerat att spelmotorn stödjer versionshantering med Git. När utvecklaren initierar Git genom UE4 skapar spelmotorn en `.gitignore` fil som refererar till mappar och filer utvecklare inte behöver versionshantera. Dessa filer genereras eller kan genereras lokalt när utvecklaren öppnar projektet i UE4. Genom UE4 kan utvecklaren även initiera *Git LFS*. När Git är kopplat till UE4 editorn kan utvecklaren direkt i gränssnittet hantera viss del av versionshanteringen. Vid ett test kunde utvecklaren direkt från UE4 skapa *commits* vid ändringar angående *Blueprints* eller 3D objekt men C++ filer kunde däremot inte inkluderas. UE4 verkade inte kunna *pusha* upp *commiten* till GitLab heller. För att inkludera C++ filer samt *pusha* upp filerna till en server fick utvecklaren fortfarande använda en terminal, Git GUI eller Git bash. Under projektet stötte projektgruppen på ett problem med att *pusha* ändringar med Git bash till GitLab på Windows efter att ha implementerat Git LSF men vid dessa tillfällen fungerade Git GUI.

H.7 Diskussion

I detta avsnitt kommer mina tankar kring resultatet med frågeställningen som utgångspunkt presenteras.

Metod

Metoden som användes för att finna den litteratur som användes i rapporten var en litteratursökning. En mer strukturerad metod som en litteraturstudie uteslöts, eftersom vid en initial testsökning hade jag svårigheter att finna artiklar som behandlade de ämnen jag sökte. Mycket av litteraturen som användes i rapporten togs därmed från officiella sidor som behandlade de koncept jag eftersökte. Informationen om hur exempelvis fil-lås för GitLab fungerade togs från GitLabs officiella sida docs.gitlab.com. Den information som användes från mina egna erfarenheter och experimentet jag utförde kommer endast från mina egna observationer och slutsatser. Detta betyder att det finns risk för mänskliga faktorer som påverkar resultatet.

Versionshantering av projekt utvecklad i spelmotorer

Hantering av binärfiler verkar vara en unik aspekt som skiljer projekt med system utvecklade i spelmotorer från andra mjukvaruprojekt. Då binärfiler för det mesta inte är läsbara för människan kommer *Git merge* vara näst intill omöjligt att utföra ifall konflikter uppstår. Eftersom när en konflikt uppstår vid en *Git merge* och det finns något i båda versionerna utvecklaren vill behålla måste utvecklaren själv undersöka filen och bestämma vad som ska sparas. När filen inte är läsbar av människan kommer utvecklaren inte kunna lösa konflikten i filen och får istället överskriva ena versionen med den andra. I projekt skapat i spelmotorer som UE4 kan detta förhindra en av de större meningarna med att kunna versionhantera. Eftersom i UE4 genereras binärfiler från exempelvis *Blueprints* och 3D scenbeskrivningar kan mycket logik och relationer hamna i binärfiler. Detta betyder att det kommer vara betydligt svårare för flera utvecklare att arbeta samtidigt utan att behöva oroa sig för att överskriva sitt eller andras arbete.

Att använda ett system som använder funktionen att låsa filer skulle kunna vara en potentiell lösning till svårigheterna med versionhantering av binärfiler eftersom de hindrar flera utvecklare att ändra samma fil. Detta kanske löser potentiella *merge* problem men det kommer fortfarande finnas en begränsning till flexibiliteten i ett projekts utveckling eftersom inte mer än en person kan utveckla i samma binärfil och potentiellt de binärfiler som är kopplade till den.

En annan aspekt att tänka på med filer som *Blueprints* genererar är att de är betydligt större än C++ filer. När två grundfiler som skapas jämfördes kunde jag se att *Blueprints* binärfil är större än de två filerna som skapas av C++ med en faktor 18. Det kan vara viktigt att tänka

på eftersom *Blueprints* kommer troligtvis vara under ständig utveckling och vid ändring av binärfiler kommer den gamla versionen sparas i historiken när den versionshanteras. Detta kanske inte har större betydelse i mindre projekt men det skulle kunna vara en faktor att ha i åtanke för större projekt då dessa gamla versioner av binärfiler kan börja komma upp problematiska storlekar. Det är också värt att tänka på att vid användning av GitLab för versionhantering kommer varje utvecklare ha en lokal version av systemet samt en gemensam kopia på en server som kan ha en storleksgräns. Vid implementation av *Git LFS* kan utvecklaren undvika att spara de gamla versioneran av binärfiler i ens *repository* men de kommer fortfarande behöva sparas på något ställe.

Git som versionshanteringsverktyg för UE4

UE4 stödjer versionhantering med Git i att det är enkelt att initiera ett projekt som ska använda Git genom UE4. UE4 har förmågan att skapa *commits* för vissa filer men verkar inte kunna inkludera filer skapade från C++ klasser. För att kunna inkludera C++ filer och lägga upp koden på en extern server fick jag gå tillbaka till att använda en terminal, Git GUI eller Git bash. När utvecklaren initierade projektet i UE4 skapades en *.gitignore* fil som redan innehöll filer som inte behövde versionshanteras. Detta tyckte jag var otroligt viktigt eftersom om utvecklaren inte redan vet mycket om filer eller är väldigt bekant med vad för filer UE4 genererar, kan det resultera i att utvecklaren börjar fylla sitt *repository* med onödig data eller data som är unik för varje session i UE4. Om detta sker kan system snabbt bli väldigt stora och börja stöta på onödiga *merge* konflikter.

Git har även många verktyg som kan hjälpa vid hantering av binärafiler. Som jag nämnt tidigare i diskussionen finns *Git LFS* som kan hjälpa med hantering av stora filer. Både *Git LFS* samt GitLab har även funktionen att låsa filer vilket kan användas för att undvika *merge* konflikter i binärfiler.

H.8 Slutsatser

I detta avsnitt kommer slutsatsen för frågeställningen presenteras.

Vilka utmaningar finns vid versionhantering av system utvecklat med spelmotorer?

Hantering av binärfiler som genereras under utvecklingen är en aspekt som är relativt unikt för system skapade med spelmotorer. Att hantera binärfiler under utveckling är både svårt och restriktivt när det gäller versionhantering då det är näst intill omöjligt att *merga* två versioner av binärfiler som har konflikter. Detta kommer att begränsa samarbetsförmågan mellan utvecklare eftersom även vid användning av fil-lås kan endast en utvecklare jobba med en binärfil. Versionshantering av binärfiler kan även bli mycket dyrt särskilt om dessa filer ändras ofta då de gamla versionerna av binärfilen kommer sparas i historiken.

Hur hanterar Git versionhanteringen av system utvecklat med Unreal Engine 4?

UE4 har implementerat stöd i deras gränssnitt för användning av Git vilket underlättar initiering av projekt för versionhantering. Vid initiering genom UE4 skapas en *.gitignore* fil som från början inkluderar filer och mappar som inte är nödvändiga att inkludera i versionhanteringsverktyget. Under initieringen kan utvecklaren även inkludera användningen av *Git LFS* vilket skapar de filer som behövs för att Git ska hantera systemet med *Git LFS*. *Git LFS* gör det möjligt för Git att hantera större filer utan att ta upp lika mycket minne i utvecklarens *repository*. Detta är möjligt eftersom *Git LFS* ersätter de stora filerna med en textfil som innehåller en pekare och sparar de faktiska filerna på en separat server. Både GitLab och *Git LFS* har även möjligheten att låsa filer vilket är ett sätt att arbeta med binärfiler där utvecklaren kan minska risken för *merge* konflikter.

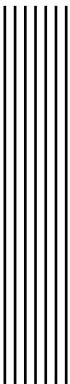


I Övriga tabeller

Tabell I.1: *Projektets kravstatus.*

Krav	Beskrivning	Prioritet	Status
1	Händelserna ska ha en allvarlighetsprioritering 1-5 där 1 är ofarligt och 5 är katastrof.	1	Ej klar
2	Händelser utan platsdata kan ska representeras i <i>direktflödet</i> men inte genom 3D-representation.	2	Klar
3	Visualiseringen av staden ska ske i 3D.	1	Klar
4	3D-representation av händelsen ska placeras ut på den virtuella kartan.	1	Klar
5	Händelser ska ha en typ, där typen beskriver händelsen med ett ord.	1	Klar
6	Händelser ska ha olika färger baserade på typ.	1	Klar
7	Det ska finnas ett <i>direktflöde</i> med händelser.	1	Klar
8	Det ska finnas ett verktygsfält där operatören kan välja olika parametrar som berör händelsernas uppvisning.	2	Klar
9	Det ska finnas ett gränssnitt för en tidslinje som operatören kan ändra historisk visning av händelser.	2	Ej klar
10	Operatören ska kunna kommentera/tagga (#) händelser i <i>direktflödet</i> .	3	Ej klar
11	Systemet ska vara skrivet i C++.	1	Klar
12	Cesium för UE4 ska användas.	1	Klar

Fortsättning på Tabell I.1			
Krav	Beskrivning	Prioritet	Status
13	Operatören ska kunna välja att göra händelser osynliga.	2	Ej klar
14	Det borde finnas en sökfunktion/filter för händelser likt toggelfunktioner.	2	Klar
15	Ska spara data historiskt så det kan använda med en tidslinje.	1	Klar
16	Ska samla in datan baserat på uppdateringar från datakällan.	2	Ej klar
17	Systemet ska ta in information från 6 valda bas-källor.	1	Klar
18	Det ska enkelt kunna läggas till flera datakällor genom att följa en manual.	2	Klar
19	Det ska samlas in återkopplingssenkäter efter varje sprint.	1	Klar
20	Det ska skrivas en manual hur systemet används.	1	Ej klar
21	Det ska skrivas en manual hur nya källor läggs till.	1	Ej klar
22	Det ska skrivas en manual hur installationen och start av systemet går till.	1	Ej klar
23	All Pythonkod måste följa formateringsstandarderna för PEP 8.	1	Klar
24	All C++ kod måste följa formateringsstandarderna enligt Visual Studios kodformaterare.	1	Klar
25	All kod ska klara alla tester i Projektets Gitlab-pipeline.	1	Klar
26	80 % av alla Python-funktioner ska ha <i>docstrings</i> .	1	Klar
27	Docstrings som skrivs följer PEP 257 – Docstring Conventions.	1	Klar



Litteratur

- [1] Discord. <https://discord.com/>. Hämtad 2022-03-07. 2022.
- [2] Facebook. *Funktioner*. <https://www.messenger.com/features>. Hämtad 2022-04-25. 2022.
- [3] InfluxData. <https://www.influxdata.com/>. Hämtad 2022-04-28.
- [4] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Hämtad: 2022-05-05. 2000.
- [5] IBM. *REST APIs*. <https://www.ibm.com/cloud/learn/rest-apis/>. Hämtad: 2022-05-23.
- [6] Microsoft. *Förstå datalagringsmodeller*. <https://docs.microsoft.com/se/azure/architecture/guide/technology-choices/data-store-overview>. Hämtad: 2022-05-22.
- [7] Python. *General Python FAQ*. <https://docs.python.org/3/faq/general.html>. Hämtad 2022-03-07. 2022.
- [8] C++. <https://isocpp.org/>. Hämtad 2022-05-06.
- [9] cplusplus.com. *Function templates*. <https://wwwcplusplus.com/doc/oldtutorial/templates/>. Hämtad: 2022-05-17.
- [10] Britannica. *SQL - definition facts*. <https://www.britannica.com/technology/SQL>. Hämtad 2022-05-23. 2022.
- [11] PostgreSQL. *About*. <https://www.postgresql.org/about/>. Hämtad 2022-03-07. 2022.
- [12] Unreal Engine. *Dokumentation Unreal Engine 4*. <https://docs.unrealengine.com/4.27/en-US/>. Hämtad 2022-05-23. 2022.
- [13] Blueprint visual scripting. <https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/>. Hämtad 2022-03-11.
- [14] Cesium. *Cesium for Unreal*. <https://cesium.com/platform/cesium-for-unreal/>. Hämtad 2022-03-09. 2022.
- [15] Visual Studio. <https://visualstudio.microsoft.com/>. Hämtad 2022-04-28.
- [16] Git. *About*. <https://git-scm.com/about>. Hämtad 2022-03-03. 2022.

-
- [17] Gitlab. *Simplify your workflow with GitLab*. <https://about.gitlab.com/stages-devops-lifecycle/>. Hämtad 2022-03-03. 2022.
 - [18] GitHub. *Git Large File Storage*. <https://git-lfs.github.com/>. Hämtad 2022-04-24.
 - [19] GitLab. *Feature branch workflow*. https://docs.gitlab.com/ee/topics/git/feature_branch_development.html. Hämtad: 2022-05-23.
 - [20] Black homepage. <https://pypi.org/project/black/>. Hämtad 2022-03-09.
 - [21] Pytest: helps you write better programs. <https://docs.pytest.org/en/7.1.x/>. Hämtad 2022-04-28.
 - [22] Flask. <https://flask.palletsprojects.com/en/2.1.x/>. Hämtad 2022-05-06.
 - [23] Scrum.org. *What is a Sprint Retrospective?* <https://www.scrum.org/resources/what-is-a-sprint-retrospective>. Hämtad 2022-04-12. 2022.
 - [24] Jeff Sutherland Ken Schwaber. *The Scrum Guide*. <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>. Hämtad 2022-06-11. 2020.
 - [25] Microsoft. *Kanban*. <https://docs.microsoft.com/en-us/azure/devops/boards/boards/kanban-overview?view=azure-devops>. Hämtad 2022-05-23. 2022.
 - [26] Trello. *Lär dig grunderna om Trello-tavlor*. <https://trello.com/sv/guide/trello-101>. Hämtad 2022-03-07. 2022.
 - [27] Upphandlingsmyndigheten. *Marknadsanalys*. <https://www.upphandlingsmyndigheten.se/inkopsprocessen/forbered-upphandling/marknadsanalys/>. Hämtad 2022-04-04.
 - [28] Google. *Google Earth Outreach*. <https://www.google.com/earth/outreach/learn/visualize-your-data-on-a-custom-map-using-google-my-maps/>. Hämtad 2022-02-23. 2022.
 - [29] RSOE. *Emergency and Disaster Information Service*. <https://rsoe-edis.org/eventMap>. Hämtad 2022-02-23. 2022.
 - [30] Esri. *Smarta Kartor*. <https://www.esri.se/sv-se/arcgis/products/smартmapping>. Hämtad 2022-02-23. 2022.
 - [31] Linköpings Kommun. <https://www.linkoping.se/open/data/vagarbete/>. Hämtad 2022-04-28.
 - [32] Linköpings Kommun. <https://www.linkoping.se/open/data/trafikmatningar/>. Hämtad 2022-04-28.
 - [33] Linköpings Kommun. <https://www.linkoping.se/open/data/cykeltrafik/>. Hämtad 2022-04-28.
 - [34] Polisen. <https://polisen.se/api/events>. Hämtad 2022-04-28.
 - [35] SMHI. <http://opendata.smhi.se/apidocs/metfcst/index.html>. Hämtad 2022-04-28.
 - [36] IBM. *Data aggregation*. <https://www.ibm.com/docs/en/tnpm/1.4.2?topic=data-aggregation>. Hämtad 2022-04-19. 2021.
 - [37] Totalförsvarets forskningsinstitut. <https://www.foi.se/>. Hämtad 2022-04-26. 2021.
 - [38] Myndigheten för samhällsskydd och beredskap. *Om MSB*. <https://www.msb.se/sv/om-msb/>. Hämtad 2022-04-26.
 - [39] Säkerhetspolisen. <https://www.sakerhetspolisen.se/>. Hämtad 2022-04-28.

- [40] Totalförsvarets forskningsinstitut. *FOI:s modell för risk- och sårbarhetsanalys (FORSA)*. Hämtad 2022-04-12. 2011. ISBN: 978-91-7056-128-3.
- [41] Myndigheten för samhällsskydd och beredskap. *Vägledning för Risk- och sårbarhetsanalyser*. 2011. ISBN: 978-91-7383-129-1.
- [42] Försvarsmakten. *Handbok Försvarsmaktens säkerhetstjänst, Informationssäkerhet*. M7739-352056. 2013.
- [43] Försvarsmakten. *Handbok Sekretessbedömning Del A*. M7739-352028. 2011.
- [44] Myndigheten för samhällsskydd och beredskap. *Säkerhet i offentlig miljö : skydd mot antagonistiska hot och terrorism, vägledning*. 2019. ISBN: 978-91-7383-973-0.
- [45] Javier Pastor-Galindo, Pantaleone Nespoli, Félix Gómez Mármol och Gregorio Martínez Pérez. "The Not Yet Exploited Goldmine of OSINT: Opportunities, Open Challenges and Future Trends". I: *IEEE Access* 8 (2020), s. 10282–10295. DOI: 10.1109/ACCESS.2020.2965257.
- [46] Maltego. <https://www.maltego.com/>. Hämtad 2022-05-19. 2022.
- [47] Google. <https://www.google.se/maps/>. Hämtad 2022-04-19. 2022.
- [48] Chris Plante. *Better with age: A history of Epic Games*. <https://www.polygon.com/2012/10/1/3438196/better-with-age-a-history-of-epic-games>. Hämtad 2022-04-29. 2012.
- [49] Epic Games. *Unreal Engine 5 is now available*. <https://www.unrealengine.com/en-US/blog/unreal-engine-5-is-now-available>. Hämtad 2022-04-29. 2022.
- [50] Epic Games. *unrealengine.com*. <https://www.unrealengine.com/en-US/>. Hämtad 2022-03-11. 2022.
- [51] Mark Ryan, Doug Hill och Dennis McGrath. "Simulation interoperability with a commercial game engine". I: *European Simulation Interoperability Workshop*. 2005, s. 27–30.
- [52] Epic Games. *Mod Authoring for Unreal Tournament 3*. <https://docs.unrealengine.com/udk/Three/UT3Mods.html>. Hämtad 2022-04-29. 2012.
- [53] Antônio Carlos A. Mól, Carlos Alexandre F. Jorge och Pedro M. Couto. "Using a Game Engine for VR Simulations in Evacuation Planning". I: *IEEE Computer Graphics and Applications* 28.3 (2008), s. 6–12. DOI: 10.1109/MCG.2008.61.
- [54] IGN Staff. *Epic Games Announces Unreal Development Kit, Powered by Unreal Engine 3*. <https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3>. Hämtad 2022-04-29. 2009.
- [55] Tom Sykes. *Unreal Engine 4 now free for academic use*. <https://www.pcgamer.com/unreal-engine-4-now-free-for-academic-use/>. Hämtad 2022-04-29. 2014.
- [56] Epic Games. *UnrealScript Language Reference*. https://docs.unrealengine.com/udk/Three/UnrealScriptReference.html#Design%20goals%20of%20_UnrealScript. Hämtad 2022-04-29. 2012.
- [57] Christian Nutt. *Epic's Tim Sweeney lays out the case for Unreal Engine 4*. <https://www.gamedeveloper.com/programming/epic-s-tim-sweeney-lays-out-the-case-for-unreal-engine-4>. Hämtad 2022-04-29. 2014.
- [58] James Lewis, David Brown, Wayne Cranton och Robert Mason. "Simulating visual impairments using the Unreal Engine 3 game engine". I: *2011 IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH)*. 2011, s. 1–8. DOI: 10.1109/SeGAH.2011.6165430.

- [59] Jérôme Leudet, François Christophe, Tommi Mikkonen och Tomi Männistö. "AILive-Sim: An Extensible Virtual Environment for Training Autonomous Vehicles". I: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. 2019, s. 479–488. DOI: 10.1109/COMPSAC.2019.00074.
- [60] Francesc Valls, Ernest Redondo, David Fonseca, Pilar Garcia-Almirall och Jordi Subirós. "Videogame Technology in Architecture Education". I: *Human-Computer Interaction. Novel User Experiences*. Utg. av Masaaki Kurosu. Cham: Springer International Publishing, 2016, s. 436–447. ISBN: 978-3-319-39513-5.
- [61] Zachary Kriz, Russell Prochaska, Cody Aaron Morrow, Cesar Vasquez, Hsingtu Wu och Rizwan-uddin. "Unreal III based 3-D virtual models for training at Nuclear Power Plants". I: *2010 1st International Nuclear Renewable Energy Conference (INREC)*. 2010, s. 1–5. DOI: 10.1109/INREC.2010.5462548.
- [62] Martin Němec, Radoslav Fasuga, Jan Trubač och Jan Kratochvíl. "Using virtual reality in education". I: *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. 2017, s. 1–6. DOI: 10.1109/ICETA.2017.8102514.
- [63] Michelangelo Scorpio, Roberta Laffi, Ainoor Teimoorzadeh, Giovanni Ciampi, Massimiliano Masullo och Sergio Sibilio. "A calibration methodology for light sources aimed at using immersive virtual reality game engine as a tool for lighting design in buildings". I: *Journal of Building Engineering* (2022), s. 103998.
- [64] Humberto Marín-Vega, Giner Alor-Hernandez, Ramón Zatarain-Cabada och M. Lucía Barrón-Estrada. "Analyzing HTML5-Based Frameworks for Developing Educational and Serious Games". I: *Technologies and Innovation ; Second International Conference, CI-TI 2016, Guayaquil, Ecuador, November 23-25, 2016, Proceedings*. 2016, s. 156. DOI: 10.1007/978-3-319-48024-4.
- [65] Daniel Camacho, Tiara Dobbs, Alessandra Fabbri, Nicole Gardner, M. Hank Haeusler och Yannis Zavoleas. "Hands on Design". I: *Intelligent and Informed - Proceedings of the 24th International Conference on Computer-Aided Architectural Design Research in Asia, CAADRIA 2019*. Vol. 1. 2019, s. 563–572. ISBN: 978-988789171-0.
- [66] M.M.M. Sarcar, K. Mallikarjuna Rao och K. Lalit Narayan. "Computer Aided Design and Manufacturing". I: PHI Learning Pvt. Ltd., 2008, s. 3–4. ISBN: 9788120333420.
- [67] Linköpings Universitet. *liu.se*. <https://liu.se/biblioteket/databaser>. Hämtad 2022-04-03. 2022.
- [68] Arvin David, Emmanuel Joy, Sathish Kumar och Sudhir John Bezaleel. "Integrating Virtual Reality with 3D Modeling for Interactive Architectural Visualization and Photorealistic Simulation: A Direction for Future Smart Construction Design Using a Game Engine". I: *2nd International Conference on Image Processing and Capsule Networks, ICIPCN 2021*. Vol. 300. 2022, s. 180–192. DOI: 10.1007/978-3-030-84760-9_17.
- [69] *Unreal Docs*. <https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Overview/>. Hämtad 2022-05-20.
- [70] Juha Ojala, Jukka Selin, Timo Partala och Markku Rossi. "Virtual Construction: Interactive Tools for Collaboration in Virtual Reality". I: *Advances in Information and Communication*. 2020, s. 341–351. DOI: 10.1007/978-3-030-39442-4_26.
- [71] Basem Eid Mohamed, Frederic Gemme och Aaron Sprecher. "Information and Construction: Advanced Applications of Digital Prototyping in the Housing Industry". I.
- [72] *Bild från 360-panorama, använd med tillåtelse från skaparen*. <https://roundme.com/tour/354437/view/1199913>. Hämtad 2022-04-28.

- [73] F. Sapienza, M. Parker, M. Bodie och S.A. Shoop. "Vehicle Modeling in Unreal Engine 4". I: *Proceedings of the 20th International and 9th Americas Conference of the International Society for Terrain-Vehicle Systems, ISTVS 2021*. 2021. ISBN: 978-194211252-5.
- [74] Razafimanjary Maminaina Aimee. *A thorough literature review of customer satisfaction definition, factors affecting customer satisfaction and measuring customer satisfaction*. Sept. 2019. DOI: 10.5281/zenodo.3483552. URL: <https://doi.org/10.5281/zenodo.3483552>.
- [75] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering". I: *ISO/IEC/IEEE 29148:2011(E)* (dec. 2011), s. 1–94. DOI: 10.1109/IEEEESTD.2011.6146379.
- [76] "IEEE Recommended Practice for Software Requirements Specifications". I: *IEEE Std 830-1998* (1998), s. 1–40. DOI: 10.1109/IEEEESTD.1998.88286.
- [77] Arne Johansson. "Marknadsaktiviteter under en innovationsprocess [Kandidatuppsats, Högskolan Väst]". Institutionen för ekonomi och it, 2008. URL: <http://hv.diva-portal.org/smash/record.jsf?pid=diva2%3A214827>.
- [78] Sai Ganesh Gunda. "Requirements engineering : elicitation techniques [Master's thesis, Högskolan Väst]". Institutionen för ekonomi och it, 2008. URL: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A215169>.
- [79] J.A. Goguen och C. Linde. "Techniques for requirements elicitation". I: *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*. 1993, s. 152–164. DOI: 10.1109/ISRE.1993.324822.
- [80] Marcus Johansson. "En studie om användbarhetskrav : hur valet av insamlingsteknik kan påverka identifieringen av olika aspekter av användbarhet [Kandidatuppsats, Högskolan i Skövde]". Institutionen för kommunikation och information, 2007. URL: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A2944>.
- [81] Zheyng Zhang. "Effective Requirements Development–A Comparison of Requirements Elicitation Techniques". I: *British Computer Society* (jan. 2007). URL: https://www.researchgate.net/publication/228717829_Effective_Requirements_Development_-A_Comparison_of_Requirements_Elicitation_Techniques.
- [82] Sema Akkas. "Kvalitetsegenskaper på en kravspecifikation [Kandidatuppsats, Högskolan i Skövde]". Institutionen för datavetenskap, 1997. URL: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A2602>.
- [83] Albert Graf och Peter Maas. "Customer value from a customer perspective: a comprehensive review". I: *Journal für Betriebswirtschaft* 58.1 (april 2008), s. 1–20. ISSN: 1614-631X. DOI: 10.1007/s11301-008-0032-8. URL: <https://doi.org/10.1007/s11301-008-0032-8>.
- [84] NASA. [Online, accessed 2022-05-23]. URL: <https://humansystems.arc.nasa.gov/groups/tlx/index.php>.
- [85] Atyanti Prabaswari, Chancard Basumerda och Bagus Utomo. "The Mental Workload Analysis of Staff in Study Program of Private Educational Organization". I: *IOP Conference Series: Materials Science and Engineering* 528 (juni 2019), s. 012018. DOI: 10.1088/1757-899X/528/1/012018.
- [86] Shengyuan Yan, Cong Chi Tran, Yu Chen, Ke Tan och Jean Luc Habiyaremye. "Effect of user interface layout on the operators' mental workload in emergency operating procedures in nuclear power plants". I: *Nuclear Engineering and Design* 322 (2017), s. 266–276. ISSN: 0029-5493. DOI: <https://doi.org/10.1016/j.nucengdes.2017.07.012>. URL: <https://www.sciencedirect.com/science/article/pii/S0029549317303436>.

- [87] Adil Ahmed, Subhash Chandra, Vitaly Herasevich, Ognjen Gajic och Brian Pickering. "The effect of two different electronic health record user interfaces on intensive care provider task load, errors of cognition, and performance". I: *Critical care medicine* 39 (juli 2011), s. 1626–34. DOI: 10.1097/CCM.0b013e31821858a0.
- [88] Chris Harrison, Gary Hsieh, Karl D.D. Willis, Jodi Forlizzi och Scott E. Hudson. "Kinetics: Using Iconographic Motion in Graphical User Interface Design". I: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, s. 1999–2008. ISBN: 9781450302289. DOI: 10.1145/1978942.1979232. URL: <https://doi.org/10.1145/1978942.1979232>.
- [89] Banbury. S. "Disruption of Office-Related Tasks by Speech and Office Noise". I: *British Journal of Psychology* 89 (1998), s. 499–517.
- [90] Pernilla Ulfvengren. "Design of Natural Warning Sounds in Human-Machine Systems". Diss. KTH, 2003. ISBN: 91-7283-656-3.
- [91] Juha Eskonen, Julen Kahles och Joel Reijonen. "Automating GUI Testing with Image-Based Deep Reinforcement Learning". I: *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. 2020. DOI: 10.1109/ACSOS49614.2020.00038.
- [92] Salma Saber, Fatma Elbadry, Hagar Negm, Rana Abu El-Ershad, Omar Magdy, Mohamed Bahnassawi, Reem El Adawi och AbdElMoniem Bayoumi. "Autonomous GUI Testing using Deep Reinforcement Learning". I: *2021 17th International Computer Engineering Conference (ICENCO)*. 2021, s. 94–100. DOI: 10.1109/ICENCO49852.2021.9715282.
- [93] PySimpleGUI. <https://pysimplegui.readthedocs.io/en/latest/>. [Online, accessed 2022-04-12].
- [94] PyAutoGUI. <https://pyautogui.readthedocs.io/en/latest/>. [Online, accessed 2022-04-12].
- [95] OpenCV. <https://docs.opencv.org/4.x/index.html>. [Online, accessed 2022-05-01].
- [96] Davide Spadini, Maurício Aniche, Magiel Bruntink och Alberto Bacchelli. "To Mock or Not to Mock? An Empirical Study on Mocking Practices". I: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 2017, s. 402–412. DOI: 10.1109/MSR.2017.61.
- [97] Kai Petersen, Peter Roos, Staffan Nyström och Per Runeson. "Early identification of bottlenecks in very large scale system of systems software development." I: *Journal of Software Maintenance and Evolution* 26.12 (2014), s. 1150–1171. ISSN: 1532-060X. DOI: 10.1002/smrv.1653.
- [98] J. Glenn Brookshear och Dennis Brylow. *Computer science : an overview*. Vol. 13. Pearson, 2020. ISBN: 9780134875460.
- [99] Ian Molyneaux. *The art of application performance testing: from strategy to tools.* O'Reilly Media, Inc., 2014.
- [100] Qi Luo, Aswathy Nair, Mark Grechanik och Denys Poshyvanyk. "FOREPOST: finding performance problems automatically with feedback-directed learning software testing". I: *Empirical Software Engineering* 22.1 (febr. 2017), s. 6–56. ISSN: 1573-7616. DOI: 10.1007/s10664-015-9413-5. URL: <https://doi.org/10.1007/s10664-015-9413-5>.

-
- [101] William W. Cohen. "Fast Effective Rule Induction". I: *Machine Learning Proceedings 1995*. Utg. av Armand Prieditis och Stuart Russell. San Francisco (CA): Morgan Kaufmann, 1995, s. 115–123. ISBN: 978-1-55860-377-6. DOI: [https://doi.org/10.1016/B9781558603776500232](https://doi.org/10.1016/B978-1-55860-377-6.50023-2).
 - [102] Perforce. *Version Control for Binary Files: Your Best Options*. <https://www.perforce.com/blog/vcs/version-control-for-binary-files>. Hämtad 2022-05-05. 2022.
 - [103] scratchapixel. *Rendering an Image of a 3D Scene: an Overview*. <https://www.scratchapixel.com/lessons/3d-basic-rendering/rendering-3d-scene-overview/rendering-3d-scene>. Hämtad 2022-04-14.
 - [104] Git. *git-merge*. <https://git-scm.com/docs/git-merge>. Hämtad 2022-05-01. 2022.
 - [105] Scott Chacon och Ben Straub. *Pro Git*. Apress, 2014.
 - [106] MSB. *Källkritik*. <https://www.msb.se/sv/amnesomraden/msbs-arbetet-vid-olyckor-kriser-och-krig/psykologiskt-forsvar/kallkritik/>. Hämtad 2022-05-21. 2021.
 - [107] J. Doboš och Steed. "3D revision control framework". I: *Web3D '12: Proceedings of the 17th International Conference on 3D Web Technology* (2012). DOI: 10.1145/2338714.2338736.
 - [108] fileinfo. *Unreal Engine Map File*. <https://fileinfo.com/extension/umap>. Hämtad 2022-04-23. 2016.
 - [109] Kevin Ring. *'Visual Scripting in Game Development*. <https://github.com/CesiumGS/cesium-unreal>. Hämtad 2022-05-02. 2022.
 - [110] Keith Engwall. *Git and GitLab in Library Website Change Management Workflows*. <https://journal.code4lib.org/articles/15250>. Hämtad 2022-05-02. 2020.
 - [111] Gitlab. *Storage usage quota*. https://docs.gitlab.com/ee/user/usage_quotas.html. Hämtad 2022-05-02.
 - [112] Brian M. Carlson. *Git LFS Specification*. <https://github.com/git-lfs/git-lfs/blob/main/docs/spec.md>. Hämtad 2022-05-22. 2019.
 - [113] GitLab. *Git Large File Storage (LFS)*. <https://docs.gitlab.com/ee/topics/git/lfs/index.html>. Hämtad 2022-05-22. 2022.
 - [114] Gitlab. *File Locking*. https://docs.gitlab.com/ee/user/project/file_lock.html. Hämtad 2022-05-04.