Project Rise

Grupp 23

Kravdokument

V.2.0

2019-05-28

Dokumenthistorik

Datum	Version	Beskrivning	Författare
190320	1.0	Skapande av kravdokument. Skrev över funktionella och kvalitativa krav.	Aevan Dino, Sebastian Viro, Muhammad Abdulkhuder
190405	1.1	Skrev om krav till tabellform. La till F3.2, F3.3, F3.4, F3.5 och IF 2.0. Skrev till i syfte. La till utförligare förklaring till Moscow modellen	Aevan Dino.
190410	1.2	La till krav F.1.1.1 - F.1.1.4.	Aevan Dino, Sebastian Viro
190429	1.3	Granskning av hela dokumentet	Rohan Samandari
190430	1.4	Uppdatering av kravs status/implementation	Seth Öberg
190508	1.5	Lagt till två krav, lagt till färgkod.	Sebastian Viro, Aevan Dino
190510	1.6	Ändrat innehållsförteckning, lagt till ny rubrik till krav (övergripande spelkrav)	Sebastian Viro
190522	1.7	Korrigering efter checklista från dokumentgranskning.	Sebastian Viro
190526	1.8	Korrigering från granskning, skrivit om krav 1.1.5, 1.2, 2.0 samt fyllt i de krav som är testade samt implementerade.	Sebastian Viro
190527	1.9	Kvalitativa- och kvantitativa krav beskrivningar, ord i lista och referenser tillagda. Sidnummer, innehållsförteckning.	Aevan Dino
190528	1.9.1	Lagt till krav F15 - F20	Sebastian Viro
190531	2.0	Lagt till flera krav samt lagt dem i rätt kategori	Muhammad Abdulkhuder.

Innehållsförteckning

Kravdokument

Syfte

Syftet med det här kravdokumentet är att specificera och sammanställa alla krav på en och samma plats. De krav som dokumenteras kommer ligga till grund för hur spelet kommer att utvecklas och kommer underlätta spårbarheten av kraven. I dokumentet definieras både funktionella så väl som icke funktionella krav och varför dessa är av betydelse för projektet. Det här dokumentet innehåller en utförlig produktbeskrivning, målgruppsbeskrivning och intressenter. I det här dokumentet kan läsaren ta reda på huruvida det är ett funktionellt eller kvalitativt krav, ta reda på om det är kravet i fråga är implementerat eller inte och slutligen även kunna se om kravet har testats. Alla krav i det här dokumentet prioriteras efter ”MoSCoW” modellen.

En beskrivning presenteras både för produkt samt målgrupp med motivation till diverse krav som tagits fram för projektet. Kraven delas upp i funktionella och icke-funktionella krav med fokus på funktionella krav till en början, en fungerande spelmekanik prioriteras över till exempelvis prestanda och tillgänglighet.

Moscow modellen

Moscow modellen används för att prioritera krav i fyra unika kategorier: Must, Should, Could, Won’t (Madsen, 2017). Modellen används huvudsakligen för att prioritera funktionella krav efter deras vikt i förhållande till varandra men också efter kundens önskan.

- **Must:** Krav som faller under ”Must” är krav som absolut måste uppfyllas för att kalla projektet lyckat.

- **Should:** Krav som faller under ”Should” är krav som bör uppfyllas. Dessa krav är inte avgörande för att kalla projektet ett lyckande men dessa krav anses meriterande för projektet.

- **Could:** Krav som faller under ”Could” är krav som hade varit till stor nytta för användare/kunden men som inte har en stor påverkan på projektet om dessa skulle utebli från projektet.

- **Wont:** Krav som faller under ”Wont” är krav som inte kommer att implementeras. Dessa har ingen större påverkan på projektet och det gör väldigt lite om dessa inte uppfylls.

Ordlista

- **MoSCoW** - En modell där krav prioriteras under följande kategorier: **M**ust, **S**hould, **Co**uld, **W**on’t.

- **Swing -** Swing är ett bibliotek i java som används av programmerare för att rita och designa grafiska gränssnitt.

- **Chanskort -** Ett kort som har en slumpmässig påverkan på spelet.

- **Allmänning -** Ett kort som likt chanskort har en slumpmässig påverkan på spelet.

- **Monopol -** Ett fastighets baserat brädspel där målet är att tvinga motspelare i konkurs genom att köpa och utveckla fastigheter. (Encyclopedia Britannica, n.d).

- **Interaktiv -** Samspel mellan datorn och användare.

Referenser

Encyclopedia Britannica. (n.d.). Monopoly. [online] Tillgänglig på: https://www.britannica.com/sports/Monopoly-board-game [Accessed 27 May 2019].

Madsen, S. 2017. How to Prioritize with the MoSCoW Technique. [online] Tillgänglig på: https://www.projectmanager.com/training/prioritize-moscow-technique

Funktionella krav

All funktionalitet som utför något beskrivs som ett funktionellt krav, t.ex. förflyttning av pjäser, tärningskast eller att sparande av information till en hårddisk. Dessa funktionella krav får inte motsäga varandra dvs krav som finns får inte ställa till med problem för varandra och all information om funktionalitet måste finnas med. Krav prioriteras efter MoSCoW modellen (se ordlista) där varje krav under varje underkategorierna kommer att prioriteras efter deras betydelse för projektet.

Färgkodning

Must	Krav i rött faller under kategorin must och måste implementeras.

Should	Krav i gult bör implementeras.

Could	Krav i blått är inte kritiska för projektet.

Testat	Krav med vit bakgrund är inte testade.

Testat	Krav med grön bakgrund är testade.

Implementerat	Vit bakgrund innebär att kravet inte är implementerat.

Implementerat	Grön bakgrund innebär att kravet är implementerat.

Krav spelbräde

F 1.0 : Spelbräde	MUST	Testat	Implementerat
Skapa en bräda. Spelbrädan ska likna en monopolspelbräda.			
Det här kravet är väldigt viktigt. Utan ett spelbräde blir projektet ett misslyckande.			

F 1.1 : Spelrutor	MUST	Testat	Implementerat
Alla rutor ska vara av olika typer. Rutorna ska delas upp i följande kategorier: Properties, Work, Fortune Teller, Tax, Go, Jail, Go to Jail, Tavern, och Sunday Church.			
Olika typer av rutor med olika events kopplade är viktigt för att få ett djup i spelet.			

F 1.1.1 : Work	MUST	Testat	Implementerat
När en spelare landar på en jobbruta (work) så skall spelaren få betalt beroende på resultatet av tärningskastet och spelarens rank.			
Jobbrutor ger spelarna mer inkomst för att hjälpa till att göra spelet snabbare. Beroende på vilken rank spelaren har ska olika mycket betalas ut. (Se appendix).			

F 1.1.2 : Property	MUST	Testat	Implementerat
När en spelare landar på en property ruta som är ledig så ska spelaren antingen kunna köpa rutan eller kunna avstå helt. Är rutan ägd av en annan spelare så ska hyra betalas ut till ägaren.			
Property rutor är väsentligt inom vårt spel.(Se appendix).			

F 1.1.3 : Tax	MUST	Testat	Implementerat
När en spelare landar på en tax ruta så måste spelaren betala skatt. Om spelaren saknar pengar så elimineras spelaren direkt. Är kravet inte implementerat så förlorar spelare. Skatten sparas i Sunday Church.			
Skatt är en del av Monopol vilket Projekt Rise drar mycket inspiration ifrån. (Se appendix)			

F 1.1.4 : Fortune Teller	MUST	Testat	Implementerat
När en spelare landar på Fortune Teller så ska ett kort dras så antingen påverkar spelaren positivt eller negativt.			
Fortune teller är inspirerad av chans -och allmäningkort från Monopol (Se appendix).			

F 1.1.5 : Tavern	MUST	Testat	Implementerat
En tavern ruta ruta kan köpas av spelare, de finns totalt 2 styck tavern rutor på spelplanen. Äger en spelare bägge tavern rutor ökar hyran (se appendix). En spelare som landar på en ägd tavern ruta måste betala till ägaren.			
Tavern är inspirerad av Monopols kraftverksrutor.			

F 1.2 : Interaktiva Rutor	SHOULD	Testat	Implementerat
En spelare ska kunna hålla över muspekaren på olika rutor för att få mer information om rutorna.			
Interaktiva rutor som visar information låter användaren få fram information under spelets gång.			

F 1.3 : Pjäser	MUST	Testat	Implementerat
Spelarna ska representeras av färgade rutor, som indikerar spelarens position på spelbrädet. Alla spelare ska ha unika färger för att undvika förvirring under spelets gång.			
Spelarna ska kunna förstå var deras pjäser befinner sig under spelets gång.			

Krav förflyttning

F 2.0 : Tärning	MUST	Testat	Implementerat
Två sex-sidiga tärningar ska ”kastas” och spelare ska kunna se tärningarna och antalet prickar de visar.			
En tärning används för att styra en spelares förflyttning.			

F 2.1 : Dubbel kast	MUST	Testat	Implementerat
Om spelaren rullar en dubbel så fördubblas förflyttningen också.			
En tärning används för att styra en spelares förflyttning.			

F 2.2 : Flytta pjäser	MUST	Testat	Implementerat
Pjäser ska kunna gå att flytta runt på spelbrädan. Spelaren **måste** ta steg som motsvarar summan av antalet prickar som två sex-sidiga tärningar visar. Tärningen ska kastas först och sedan ska spelaren röra sig framåt.			
det är väsentligt för spelet att pjäserna flyttar sig efter antalet steg.			

F 2.2.1 : Animering	SHOULD	Testat	Implementerat
Förflyttningen skulle kunna vara animerad alltså den ska visa processen.			
Animerat förflyttning skulle göra det enklare för spelaren att hänga med.			

F 8.0 : Avsluta tur	MUST	Testat	Implementerat
När en spelare känner sig färdig med sin tur ska hen kunna avsluta sin tur. En spelare måste åtminstone kastat tärning för förflyttning.			
När en spelare är klar med sin runda ska de kunna klicka på en "Avsluta tur" knapp för att nästa spelare ska kunna påbörja sin runda. Krav för att en spelare ska kunna avsluta sin runda är att de måste ha slått tärningen, flyttat sig och gjort andra nödvändiga steg då de landat på en ruta (tex betala rätt mängd till den som äger en fastighet).			

Krav Ranksystem

F 3.0 : Ranka upp	MUST	Testat	Implementerat
Spelare ska kunna klättra i samhällspyramiden. När en spelares förmögenhet ökar (antal mynt och antal bostäder) ska också hans rank uppdateras.			
Det här är viktigt för spelet då det är det här som skiljer Rise från vanlig Monopol. Se appendix för gränser för ranker.			

F 3.1 : Ranka ner	COULD	Testat	Implementerat
Skulle en spelare tappa i förmögenhet (wealth) så ska hen flytta ner en rank.			
Det här kan bli svårt att hålla koll på men definitivt något att försöka få gjort.			

F 3.2 : Rank Peasant	MUST	Testat	Implementerat
Första ranken, här alla spelare börjar. En spelare på ranken är begränsad på hur mycket dem kan uppgradera sina fastigheter samt hur mycket dem kan få utbetalt från work rutor.			
Lägsta ranken med flest begränsningar, se appendix för exakta begränsningar.			

F 3.3 : Rank Knight	MUST	Testat	Implementerat
Andra ranken, ökad utbetalning vid jobb och kan uppgradera sina fastigheter ytterligare.			
Andra ranken, mer funktioner av spelet låses upp vilket skjuter spelet framåt.			

F 3.4 : Rank Lord	MUST	Testat	Implementerat
Näst högsta ranken, full utbetalning från work rutor och kan uppgradera fastigheter fullt.			
Ranken innan vinst, med alla funktioner maximalt upplåsta så blir det förhoppningsvis mer intensivt mot slutet av spelet.			

F 3.5 : Rank King	MUST	Testat	Implementerat
Högsta ranken, den spelare som först når hit vinner spelet.			
Spelets vinstvillkor.			

F 3.6 : Ändra pjäser	COULD	Testat	Implementerat
En spelares spelpjäs ska indikeras av en symbol. När en spelare rankar upp/ner så ska denne spelares pjäs ändras till rätt schackpjäs. **(Peasant = bonde, Knight = springare, Lord = löpare, Ruler = kung)**			
Det är till för att underlätta för användaren att se sin rank och position på spelbrädet.			

F 15.0 : Eliminera spelare	Must	Testat	Implementerat
När en spelare har 0 eller mindre gold coins vid en tidpunkt i spelet så ska spelaren elimineras.			
Spelet måste ha ett sätt att eliminera spelare.			

Krav bakgrundsmusik

F 4.0 : Bakgrundsmusik	SHOULD	Testat	Implementerat
Det ska kunna gå att spela musik i bakgrunden av spelet. Användaren ska kunna stänga av och sätta på musik under spelets gång.			
Det här behövs för att skapa en skön atmosfär, idealt medeltida musik eller något liknande som passar spelets tema.			

Krav dialoger

F 5.0 : Köpdialog	SHOULD	Testat	Implementerat
När en spelare hamnar på en fastighet så ska spelaren få en dialog som ger spelaren valet att köpa fastigheten.			
Det här ger spelet ett bättre utseende och är dessutom definitivt något som hade gjort spelet mer användarvänligt.			

F 5.1 : Köpa fastigheter	Must	Testat	Implementerat
En spelare ska kunna köpa oägda fastigheter.			
Fastigheterna på brädet måste kunna tilldelas spelare på något sätt.			

F 5.2 : Säljdialog	SHOULD	Testat	Implementerat
Spelare ska kunna välja att sälja sina fastigheter under spelets gång.			
Det här kommer till nytta när spelare får ont om pengar.			

F 5.3 : Sälja fastigheter	Should	Testat	Implementerat
En spelare ska kunna köpa sälja sina fastigheter. Fastigheter säljs till inköpspris.			
En spelare ska kunna sälja sina ägda fastigheter.			

F 5.2 : Bytesdialog	COULD	Testat	Implementerat
Spelare ska kunna välja att byta fastigheter och gold coins med varandra under spelets gång. Det här ska kunna göras i ett separat fönster eller på startskärmen.			
Det här hjälper till att skapa rivalitet mellan spelare men också för att kunna ge spelare möjligheten till att uppnå monopol.			

F 5.3 : Auktionsdialog	COULD	Testat	Implementerat
Spelare ska kunna lägga sina fastigheter upp på aktion, och den spelare som bjuder högst summa är den som får huset om ägaren accepterar. En eliminerad spelares fastigheter ska även läggas ut på auktion.			
Med auktion kan en spelare försöka få mer än inköpspris för sina fastigheter. En eliminerad spelares fastigheter som läggs ut till auktion skapar en intressant situation för de spelare som är kvar.			

F 5.4 : Dialogruta property -oägd	Could	Testat	Implementerat
En dialogruta ska visa sig när en spelare landar på en property ruta som ingen äger.			
Dialogrutan ska hjälpa spelaren göra val vid hantering av oägda fastigheter.			

F 5.5 : Dialogruta property - ägd	Could	Testat	Implementerat
En dialogruta ska visa sig när en spelare landar på en property ruta som en annan spelare äger.			
Dialogrutan ska visa hur mycket en spelare ska betala i hyra.			

Fastighet Krav

F 18.0 : Köpa Taverns	Must	Testat	Implementerat
En spelare ska kunna köpa oägda taverns.			
De två tavern rutorna på brädet måste kunna tilldelas spelare på något sätt.			

F 19.0 : Sälja taverns	Should	Testat	Implementerat
En spelare ska kunna sälja sina taverns. Taverns säljs till inköpspris.			
En spelare ska kunna sälja sina tavern rutor.			

F 16.1 : Visa fastigheter	Must	Testat	Implementerat
En spelare ska kunna se sina ägda fastigheter			
Fastigheterna på brädet måste kunna tilldelas spelare på något sätt.			

F 20.0 : Uppgradera fastigheter	Should	Testat	Implementerat
En spelare ska kunna uppgradera sina fastigheter. Fastigheter börjar på level 0 och kan uppgraderas som mest till level 5. Hur mycket en spelare kan uppgradera beror på en spelares rank (se appendix).			
Fastigheterna på brädet ska kunna uppgraderas av sina ägare.			

F 20.1 : Nedgradera fastigheter	Should	Testat	Implementerat
En spelare ska kunna nedgradera sina ägda fastigheter.			
Fastigheterna på brädet ska kunna nedgraderas av sina ägare.			

Krav Startup

F 9.0 : Starta spel	MUST	Testat	Implementerat
Spelare ska kunna starta spelet lokalt på sin dator.			
Det måste gå att starta spelet.			

F 9.1 : Start screen	MUST	Testat	Implementerat
Vid start av spel ska en start screen visa sig vilket låter spelare registrera sig.			
Efter start av spelet ska flera spelare kunna registrera sig via en startruta och både kunna välja namn och färg.			

F 10.0 : Registrering av spelare	MUST	Testat	Implementerat
Flera spelare ska kunna registrera sig till spelet. Spelare ska kunna välja namn samt färg.			
Eftersom spelet körs lokalt så ska flera spelare kunna ansluta sig till en session av spelet.			

Krav övergripande

F 11.0 : Historik	MUST	Testat	Implementerat
Händelser som sker under spelets gång ska visas i ett historikfönster.			
Historik underlättar för spelare att se vad som händer/har skett i spelet.			

F 12.0 : Fusktärning	Should	Testat	Implementerat
En fusktärning som låter en användare mata in antal steg hen önskar att förflytta sig.			
Fusktärning underlättar testning processen.			

Krav PVP

F 6.0 : PVP nätverk	COULD	Testat	Implementerat
Göra det möjligt för spelare att möta varandra över olika datorer.			
Nätverk möjliggör att spelare kan möta varandra från sina egna datorer.			

F 7.0 : Information	COULD	Testat	Implementerat
Ge användare information, tutorial och tips om spelet.			
Kan vara bra för nybörjare.			

Kvalitativa krav

Kvalitativa krav är krav som baseras på en viss kvalitet eller egenskap snarare än på en viss kvantitet eller ett uppmätt värde. Kvalitativa krav är därför betydligt svårare att verifiera och validera.

IF 1.0 : Portabilitet	**Could**	**Testat**	**Implementerat**
Spelet ska kunna gå att spela på skärmar av olika upplösningar.			
Det här är viktigt därför att alla inte har samma upplösning på sina skärmar.			

IF 2.0 : Programspråk	**MUST**	**Testat**	**Implementerat**
Spelet ska skrivas i Java.			
Det är det enda språket som gruppen är bekant med.			

IF 3.0 : Estetik	**MUST**	**Testat**	**Implementerat**
Använda Java Swing för att designa användargränssnittet.			
Det här är viktigt därför att alla inte har samma upplösning på sina skärmar.			