Professionshøjskolen UCN

IT-Uddannelserne

Datamatiker

DMA-CSD-V24, Gruppe 2

Vejleder; Karsten Jeppesen

Realtidsauktionshus

Systemudvikling

Antal anslag: 46.903

Antal normalsider: 19.5

Magnus Tomra Engberg
Oscar Seistrup Herman
Kasper Mikkelsen
Matias Holm Nielsen
Jeppe Bøss Sørensen

23-05-2025

Abstract

This report documents the development of a real-time auction house by Group 2, Computer Science, UCN. The objective was a distributed system for fair bid handling with instant updates. Development methodology choices (SCRUM, XP, Kanban) were analyzed against plan-driven development using Boehm Turner's model. An agile approach, based on SCRUM and XP practices like Pair Programming and TDD, was selected. Quality assurance and risk management were central. The architecture is a 3-tier system (web app, API, desktop client). The reflection details learning with agile methods, challenges with user stories and "Definition of Done," and successes with XP. It was concluded that an agile "SCRUM-but" model was most appropriate. The project provided valuable practical experience.

Indledning

Denne rapport dokumenterer systemudviklingsprocessen for et "Realtidsauktionshus", et projekt udført af Gruppe 2 på Datamatikeruddannelsen UCN. Rapporten vil belyse, hvordan gruppen har tacklet denne udfordring.

Baggrunden for projektet er behovet for at designe og implementere et realtidsauktionshus, der kan håndtere de krav, der stilles til online auktionsplatforme. Relevansen af et sådant emne er høj, da realtidssystemer, der sikrer retfærdighed, effektivitet og øjeblikkelig brugerfeedback, er centrale i mange digitale løsninger. Udfordringen ligger specifikt i at udvikle et distribueret system, der kan håndtere samtidige bud og levere øjeblikkelige opdateringer. Derfor lyder vores problemformulering således:

Problemformulering

Hvordan kan vi udvikle et distribueret realtidsauktionshussystem, der sikrer retfærdig og effektiv håndtering af samtidige bud, samt leverer øjeblikkelige opdateringer til alle brugere?

Underspørgsmål til systemudvikling

Hvilken udviklingsmetode og proces kan bedst sikre, at krav om realtidskommunikation, stabilitet og retfærdige bud bliver opfyldt i projektforløbet?

Teoretisk Baggrund

Agile Arbejdsmetodiker

Dette afsnit vil omhandle den teoretiske side af de agile arbejdsmetodik og hvordan de forskellige værdier, roller, events og artefakter spiller sammen.

SCRUM

Scrum er baseret på empirisk proceskontrol og lean tankegang, dvs. at mange af ens beslutninger kommer fra erfaring hvor man også fokusere på det væsentlige og på at reducere spild. Scrum er en iterativ tilgang hvor man forsøger at optimere forudsigeligheden og samtidig prøver at kontrollere risici. Det egner sig bedst til mindre grupper, hvor gruppens færdigheder samlet set dækker de behov der skal til for at udføre opgaven eller projektet. Det hele bliver samlet i scrum events hvor man tager højde for værdier, så kaldte “scrum-pillars” og accountabilities. [1] (s. 3)

Scrum pillars

Scrum pillars omhandler gennemsigtighed, inspektion og tilpasning.

Her er gennemsigtighed det at processen og resultatet skal være klart og synligt for alle medlemmer i gruppen. Lav gennemsigtighed kan føre til værdien på produktet mindsker og ens risici bliver øget.

Inspektion handler om at de forskellige scrum artefakter og aftalte mål skal inspiceres ofte for at opdage potentielle afvigelser eller problematikker med projektet. Derfor er det også vigtigt at for at man kan inspicere, så skal man have gennemsigtighed.

Sidst har vi tilpasning, som er hvis en proces afviger fra grænserne eller hvis produktet ikke er acceptabelt, skal der justeres nogle ting. Den justering skal foretages hurtigst muligt for at minimere yderlige afvigelser. For at man kan tilpasse sig, bliver man nemlig nødt til at inspicere de forskellige artefakter løbende. På den måde kan man se hvordan de tre pillars hænger sammen og hvorfor de er vigtige i processen. [1] (s. 4)

Scrum værdier

Udover de tre pillars, så har scrum også fem værdier man inden for scrum prøver at efterleve. Det er engagement, fokus, åbenhed, respekt og mod. Når man arbejder i et scrum team, forpligter man sig til at nå sine mål, til at støtte hinanden i teamet og til at opnå den bedst mulige fremdrift. Det er hvad de fem værdier skal sikre.

Man skal være engageret og have fokus på projektet og udfordringerne. Samtidig skal man også være åben og have mod til f.eks. at sige højt hvis man har nogle udfordringer man ikke selv kan løse, og der skal de andre medlemmer have respekt og hjælpe til. Alle de fem værdier er også med til at skabe tillid i teamet og skaber en endnu stærkere relation, som fremmer fremgang mere. [1] (side 4)

Acountabilities

Hvis man kigger på scrum teamet, så har er der 3 acountabilities, developers, product owner og scrum master. Hver især har forskellige ansvarsområder, men samlet har de et mål, som er product goal. I et scrum team er alle selvstyrende, hvilket betyder at internt beslutter de selv hvad, hvornår og hvordan de gør. Typisk er et team lille nok til at forblive agil, og samtidig stort nok til at få udført et stykke arbejde inden for et sprint. Det vil ofte være 10 eller færre personer, da man generelt kommunikerer bedre og er mere produktive i små teams, og man bør derfor overveje hvis teamet er for stort, at dele det op i 2 teams, såfremt man har ressourcer til det.

Developers er de personer i teamet som har ansvar få at skabe hver increment i hvert sprint. Det betyder at det er dem der skaber det faktiske produkt. Derudover skal de også udarbejde en plan for hvert sprint, kaldet en sprint backlog, samt det de producere skal overholde definition of done.

Product owner er ansvarlig for at maksimere produktets værdi baseret på resultatet af hele scrum teamets arbejde. Det er product owneren der formidler product goal, laver og prioritere product backloggen, samt synliggør den, så hele teamet forstår det. Det er bl.a. også product owneren der står for kommunikationen mellem virksomheden og scrum teamet.

Sidst er der scrum master, der er ansvarlig for at teamets effektivitet igennem den agile arbejdsmetodik, scrum. Scrum master hjælper teamet med at holde fokus på det der skaber værdi, at definition of done bliver overholdt og fjerner evt. hindringer der kan komme i løbet af sprints. [1] (s. 5-7)

Scrum Events

Scrum har fem events. For hvert sprint kommer man igennem alle events. Disse events er designet til at muliggøre gennemsigtigheden og dermed også mulighed for at inspicere og tilpasse sig.

Først har man sprints. Sprints er hovedeventet i scrum, hvor alle de andre events sker inde i et sprint. Hvert sprint har altid en fast og samme længde, typisk mellem to til fire ugers længde. Alt det man laver i et sprint, er for at nå et samlet mål, nemlig sprint goal. Det er altså her man udvikler på ens produkt.

Dernæst har man sprint planning, som er det først man foretager sig når man starter et sprint. I sprint planning, får man kortlagt hvad man skal lave i løbet af hele sprintet, så hele scrum teamet kender planen. Man har overordnet tre emner ved et sprint plannings møde, “Hvorfor er dette sprint værdifuldt?”, “Hvilken færdig funktionalitet leveres i dette sprint?” og “Hvordan vil det valgte arbejde blive udført?”. Her laver man et sprint goal for hele sprintet hvor man i teamet tager items fra product backlog og flytter dem over i sprint backloggen. Det er developerne der planlægger hvor meget de kan nå ud fra tidligere erfaring.

Daily scrum er et daglig 15-minutters event for developers, som har til formål at inspicere fremskridt mod sprint goal og hvor man får mulighed for at justere på sprint backlog. Eventet sker altid samme sted og samme tidspunkt for at gøre det mindre kompleks.

Næstsidst i et sprint, har man sprint review. Det er et event hvor man som scrum team og sammen med interessenter, viser resultatet af det produkt man har formodet at levere i løbet af sprintet. Her får interessenterne mulighed for at komme med nye ændringer som så skal tilpasses i product backloggen. Det er derfor vigtigt at sprint review ikke bare bliver en præsentation fra developers, men derimod en produktiv samarbejdssession.

Sidst i et sprint har man sprint retrospective, som har til formål, at snakke om hvordan man øger kvaliteten og effektiviteten. Dvs. man kigger altså ikke på produktet, men på den enkle person og hvordan processen har været i løbet af sprintet. Her snakker man om hvad der gik godt, hvad der gik mindre godt og hvordan man evt. fik løst de problemer man stødte ind i, i løbet af sprintet. [1] (s. 7-10)

Scrum Artifacts

Inden for scrum har man 3 artefakter, product backlog, sprint backlog og increment. Fælles for dem alle, er at de repræsenterer et stykke arbejde eller værdi. Hvert artefakt har en forpligtelse til at sikre, at den giver information til at fremme gennemsigtighed.

Product backlog er en dynamisk, prioriteret liste over hvad der nødvendigt af user stories for at lave et produkt. Det er her hvor scrum teamet kan se hvilket arbejde der skal udføres. Det er fra denne, scrum teamet tager fra til sprint plannings og putter over i sprint backloggen. Målet med at have en product backlog, er at alle har hvad der skal bruges for at få udført hvad der kræves til at opfylde product goal.

Sprint backloggen består af et sæt af items, man har taget fra product backloggen, der er valgt til det enkle sprint. Det er derfor en handlingsplan til hvad man regner med at kunne levere af increment inden for et sprint. Det er developers der planlægger hvad og hvordan sprint backloggen ser ud. Man kan hver dag i daily scrum revurdere og opdatere sprint backloggen. Formålet med sprint backloggen er at kunne opfylde ens sprint goal. Det er det eneste mål man har for et sprint, og er noget man planlægger i starten af et sprint under sprint planning.

Sidst er der et increment. Det er et konkret trin mod ens product goal. Hvert increment er et stykke færdigt arbejde der tilføjes på tidligere increment, som til sidst ender ud i det færdige produkt. Måden man vurderer et increment, er at man laver en definition of done. Det er en række krav ens increment skal opfylde før man altså kan sige at man er færdig med et increment. Definition of done skaber gennemsigtighed ved at give alle en fælles forståelse af det arbejde som kræves for at afslutte et item i product backloggen. Hvis organisationen ikke har en standard for definition of done, så skal man som scrum team selv oprette sådanne en. [1] (s. 10-12)

XP

eXtreme Programming, eller herefter betegnet XP, er en agile arbejdsmetodik specificeret til systemudvikling. XP er letvægts arbejdsmetode, beregnet til små til mellem størrelse teams, der udvikler software i en verden hvor vage og hurtigt skiftene krav, til systemet, er en realitet[2] Arbejdsmetoden og navnet er baseret på ideen om at hvis noget virker, i forbindelse med at udvikle software, skal man gøre det ekstremt meget. XP blev første gang foreslået af ingeniør og programmør Kent Beck, i 1996. Den nyeste

version er bygget op omkring 5 kerneværdier, 14 principper og 24 praksisser [3]. I dette afsnit gennemgår vi de 5 kerneværdierne og de 13 vigtigste praksisser, og hurtigt skitsere fordele og ulemper ved denne arbejdsmetodik.

De 5 XP kerneværdier

Kommunikation

God kommunikation er kritisk for at projektet bliver en succesfuldt, og mangel på samme afføder en masse problemer. I et lille til mellem størrelse hold, der skal kode sammen og modtage løbende feedback fra kunden/Product owneren, er det vigtigt at folk kan kommunikere ordentligt med hinanden, så der ikke opstår unødige konflikter eller misforståelser. Desuden bygger XP på nogen praksisser, som kræver god kommunikation, hvis de skal have det ønskede resultat. [4] (slide 13)

Simplicitet

Simplicitet er også en kerneværdi, da simpel opbygning af et system fx. Gør det markant nemmere at lave ændringer hurtigt. Derfor vil man i XP hellere gøre tingene så simpelt så muligt i dag, og gøre noget ekstra i morgen for at ændre det, hvis behovet er der. [4] (slide 14)

Feedback

Feedback er helt centralt for XP-processen. Både under planlægning/skitsering af systemet, og under udviklingen, er processen bygget op omkring løbende feedback, på alle niveauet. Fx. Internt på udviklerholdet eller imellem kunden og udviklerne. Hvis der ikke benyttes feedback og kommunikeres ordentligt, på alle niveauetter, falder hele tanken om en let agil arbejdsproces til jorden. [4] (slide 15)

Mod

Mod er også en forudsætning for, at de andre kerneværdier, principper og praksisser kan fungere. I XP skal man have mod til at eksperimentere, afvise ting, tage beslutninger og generelt kommunikere det man mener. [4] (slide 16)

Respekt

Udpenslet som en værdi i den nye version af XP. Respekt er egentlig også en underliggende forudsætning for at alle de andre kerneværdier, kan komme i spil. Det omhandler gensidig respekt internt på holdet og generelt imellem alle der er inde over processen. Det sikrer et godt arbejdsmiljø og promovere fx. God kommunikation, saglig feedback og modet til at ytre sine meninger. [4] (slide 17)

XPs 13 vigtige praksisser

Sit Together

XP anbefaler, at hele teamet sidder samlet i et fysiske rum. Det fremmer hurtig og god kommunikation, så spørgsmål kan afklares med det samme, og problemer løses hurtigt. Det styrker også relationerne i teamet og gør det nemt at udveksle erfaring på tværs af specialer. [4] (slide 20)

Whole Team

Alle roller og kompetencer, der er nødvendige for at levere software, skal være tilgængelige i teamet, fra udviklere, testere til product owner. Det betyder, at teamet har al den viden og beslutningskraft, der skal til for at arbejde selvstændigt og effektivt, uden at skulle vente på eksterne aktører eller konsulenter. [4] (slide 21)

Informative Workspace

Arbejdsmiljøet skal være informativt, så alle hurtigt kan få overblik over projektets status. Det gøres fx med visuelle værktøjer som whiteboards, burndown charts, task boards eller digitale dashboards. Det skaber fælles forståelse og gør det nemt at opdage flaskehalse eller problemer i tide, og derved effektivisere processen og implementere ændringer. [4] (slide 22)

Energized Work

XP tror på, at mennesker arbejder bedst, når de er veludhvilede og motiverede. Derfor undgås overarbejde, og der arbejdes i et bæredygtigt tempo, hvor fokus er på høj kvalitet frem for mange timer. Energisk arbejde handler om at være fokuseret og produktiv i løbet af normale arbejdstider. [4] (slide 23)

Pair Programming

To udviklere arbejder sammen foran samme computer, hvor én skriver kode, og den anden overvåger og støtter. Dette sikrer konstant feedback, højere kodekvalitet og fælles ejerskab over koden. Det fremmer også læring og opbygger fællesskab og fælles forståelse, for fx problemer eller kodestruktur i teamet. [4] (slide 24)

Stories

User-stories beskriver funktioner fra brugerens synspunkt og bruges til at forstå og diskutere kravene til systemet. De hjælper med at nedbryde opgaver i mindre bidder og gør det lettere at prioritere hvilken funktionalitet der giver kunden mest værdi. Historier er korte, konkrete og fungerer som samtalestartere mellem team og kunde. [4] (slide 25)

Weekly Cycle

XP arbejder i ugentlige iterationsforløb, hvor teamet planlægger, udvikler og demonstrerer fungerende software. Hver uge afsluttes med feedback og evaluering, hvilket sikrer hurtige tilpasninger og løbende forbedringer. [4] (slide 26)

Quarterly Cycle

Ud over de korte iterationsforløb arbejder XP også med kvartalsvise planlægningsmøder. Her tages der et skridt tilbage for at vurdere projektets overordnede retning og justere mål og prioriteter. Det sikrer, at teamet stadig arbejder på det, der skaber størst værdi for kunden. [4] (slide 27)

Slack

I XP planlægges der med ekstra plads til uforudsete, små lavprioritets opgaver eller forbedringer. Dette "slack" betyder, at teamet ikke er fuldt booket og har mulighed for at håndtere fejl, læring eller forsinkelser i tidsplanen. Det giver fleksibilitet og hjælper med at undgå stress og overbelastning i travle perioder. [4] (slide 29)

Ten-Minute Build

Det samlede system skal kunne bygges og testes automatisk på under ti minutter. Det gør det muligt hurtigt at få feedback på ændringer og sikrer, at problemer opdages tidligt. Det er afgørende for effektiv kontinuerlig integration og produktivitet. [4] (slide 30)

Continuous Integration

XP anbefaler, at kode integreres i development branch ofte, gerne flere gange dagligt, og testes automatisk ved hver integration. Det mindsker risikoen for konflikter og gør det nemmere at finde fejl hurtigt. Continuous Integration understøtter et stabilt udviklingsmiljø, og gør det nemmere at samarbejde. [4] (slide 31)

Test-First Programming

Også kendt som Test Driven Development, her skrives testene, før man skriver koden, der skal opfylde dem. Det tvinger udvikleren til at tænke over krav og design, inden implementeringen går i gang. Test-first fører til mere robust, testbar og velstruktureret kode, samtidig med at det fungerer som dokumentation. [4] (slide 32)

Incremental Design

XP fokuserer på gradvis og løbende forbedring af systemets design. I stedet for at forsøge at lave et komplet design fra starten, justeres og forbedres designet løbende i takt med nye behov opstår. Det gør systemet mere fleksibelt, lettere at vedligeholde og udvikle på lang sigt. [4] (slide 34)

Kanban

Kanban er så tæt på kaos programmering som man kan komme, uden at arbejde i direkte kaos. Det minder om SCRUM men er meget forsimplet, det er også kaldet JIT(just in time) udvikling. Kanban hjælper med at visualisere workflowsne ved at lave et kanban-board. Ved at bruge boardet sætter man en aftalt flaskehals på hvor mange opgaver der kan være i gang på samme tid. Kanbans flexibilitet og simplicitet gør at det kan benyttes med andre metodikker, som scrum, uden at forstyrre dets metoder. Det kan bruges til at udføre grundigt planlagte projekter eller meget simple opgaver. Kanban board kan også bruges både med eller uden tidestimater, hvilket gør at det udskiller sig fra andre metodiker, det er også derfor det læner sig meget op af kaos. [5]

SCRUM, XP og Kanban, kan bruges i tandem og er agile arbejdsmetoder med højere eller mindre grader af kaos. De deler fokus på kommunikation, fleksibilitet, løbende planlægning og hyppig feedback under udvikling af systemer.

Plan-dreven vs. agil udvikling

I dette afsnit kigger vi på de to dominerende tilgange til systemudvikling: den plan-drevne og den agile. Hvor plan-dreven udvikling er karakteriseret ved en lineær og forudsigelig proces med faste faser og dokumentation (fx Waterfall-modellen), fokuserer agil udvikling på fleksibilitet, løbende tilpasning og tæt samarbejde med brugeren/kunden (fx Scrum og Kanban). Tilgangenes forskelle og anvendelsesområder bliver uddybet med afsæt i relevant teori, herunder Boehm & Turners ramme for at vurdere, hvornår hvilken metode egner sig bedst. Målet er at skabe en forståelse for, hvornår og hvorfor man vælger én tilgang frem for den anden i praksis.

Overordnet tilgang

For at tydeliggøre forskellene mellem plan-dreven og agil udvikling viser tabellen nedenfor en sammenligning af centrale egenskaber ved de to tilgange. Vigtigt at bemærke at den Agile tilgang, forsøger at tage feedback og ændringer i planen løbende, samt at det forventes. Hvor Plandreven er mindre fleksibel og kræver gode og detaljerede planlægningsfaser, før tidsplaner kan overholdes.

Egenskab	Unified Process (Plan-dreven)	Agile (Scrum, XP)
Tilgang	Vægt på planlægning og dokumentation.	Vægt på fleksibilitet og samarbejde.
Proces	Iterativ og inkrementel, men med tung planlægning.	Iterativ og inkrementel med korte cyklusser (sprints).
Forandringer	Svære at håndtere sent i processen.	Forventes og integreres løbende.
Kundeinvolvering	Primært i starten og slutningen.	Løbende samarbejde og feedback.

Struktur og roller

Forskellige udviklingstilgange organiserer teams og roller på forskellige måder. Tabellen her viser, hvordan rollerne fordeler sig i henholdsvis den plan-drevne Unified Process og i agile metoder som Scrum og XP. Plan-dreven udvikling har ofte klart adskilte og specialiserede roller, mens agile metoder lægger vægt på tværfaglige teams og tæt samarbejde mellem rollerne, ofte med kunden som en aktiv del af teamet.

Egenskab	Unified Process (Plan-dreven)	Scrum (Agil)	XP (Agil)
Roller	Arkitekt, udvikler, tester, projektleder osv.	Product Owner, Scrum Master, Scrum Team.	Programmer, Coach, Kunde (on-site), Tester osv.

Dokumentation og kode

Tabellen nedenfor belyser forskellene mellem plan-dreven og agil udvikling med fokus på dokumentation, kodekvalitet og standarder. Hvor Unified Process lægger vægt på grundig dokumentation og faste kvalitetskontroller, prioriterer agile metoder som Scrum og XP løbende test, minimal dokumentation og samarbejde om koden for at sikre høj kvalitet og hurtig levering.

Egenskab	Unified Process	Agile
Dokumentation	Omfattende (UML, use cases, design dokumenter).	Kun det nødvendige – "working software over comprehensive documentation".
Kvalitetssikring	Testdesign og review i faste faser.	Kontinuerlig test og refaktorering.
Kodestandarder	Typisk defineret fra start.	XP har meget fokus på kodestandarder, TDD, parprogrammering.

Feedback og tilpasning

Nedenstående tabel sammenligner, hvordan Unified Process og agile metoder som Scrum og XP håndterer feedback og tilpasning undervejs i udviklingsprocessen. Fokus er især på, hvor hyppigt feedback indhentes, hvor let det er at tilpasse sig ændringer, og hvordan risiko håndteres i de to tilgange.

Egenskab	Unified Process	Agile (Scrum, XP)
Feedback-frekvens	Lav – typisk efter hver fase.	Høj – efter hver iteration eller dagligt.
Tilpasningsevne	Lav til medium – ændringer er dyre.	Høj – ændringer er forventede og velkomne.
Risiko	Identificeres tidligt, men ændringer sent er svære.	Mindre risiko pga. hurtig feedback og hyppige leverancer.

Unified Process følger en mere struktureret og dokumentationstung proces, hvor det er vigtigt at gøre tingene rigtigt første gang, da ændringer eller afvigelse fra den originale plan, er ressourcetungt at gøre. Plandreven er især velegnet i projekter med høje krav til stabilitet og forudsigelighed. Denne tilgang fungerer særligt godt, når kravene er veldefinerede fra starten, og risikoen for ændringer undervejs er lav, eksempelvis i længerevarende projekter med klart definerede mål og faste rammer. Projekter med et kvalitetskriterie baseret på "*Right the first time*".

Agile metoder som Scrum og XP er mest effektive når kvalitetskriteriet er "*fit for purpose"*. Processen tilpasses løbende brugerens behov og ændringer i omgivelserne. Her er fleksibilitet, hyppig feedback og kontinuerlig forbedring centrale elementer, hvilket gør dem særligt velegnede i projekter med høj usikkerhed, hurtigt skiftende krav og tæt samarbejde med kunden. Som fx projekter hvor man skal udvikle eller afsøge nye løsninger undervejs.

Denne kontrast afspejler også Boehm & Turners pointe om, at valget mellem plan-dreven og agil ikke bør være ideologisk, men pragmatisk. Metodevalg bør være afhængigt af konteksten og projektets karakter. Mange moderne virksomheder anvender derfor en hybrid tilgang fx. "SCRUM BUT", hvor elementer fra flere metoder kombineres eller håndplukkes for at opnå både struktur og fleksibilitet.

Boehm Turners model

Boehm Turner modellen er en model vi har brugt i vores projekt for at identificerer den bedst mulige udviklings metode. Denne model er meget nyttig, da den tydeligt illustrerer hvordan projektets egenskaber fordeler sig mellem agil og plandrevet projektstyring, ud fra de fem centrale dimensioner. De fem dimensioner er Personnel, Dynamism, Culture, Size og Criticality. [6]

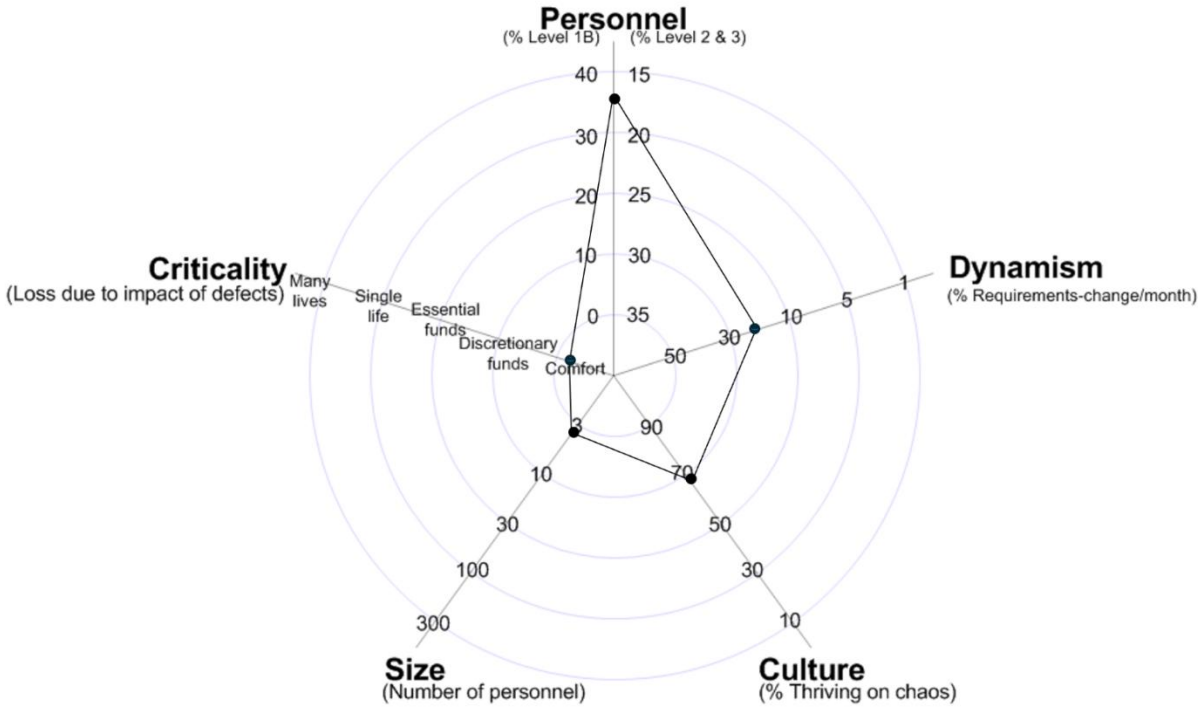
Personnel omhandler hvor høje kompetencer personerne i teamet har. Her har vi vurderet at vi alle ligge omkring det der hedder 1B og måske 1A (se bilag 1), da vi alle er studerende, men har lært en masse indtil nu. Dette passer meget godt ind i plandrevet udvikling, hvor der er klare rammer for hvad hver især skal lave.

Dynamism handler om hvor meget kravene til systemet ændrer sig på måneds basis. Hvis kravene ændrer sig meget, så er agil helt klart det rigtige valg. Her har vi valgt at sige at i og med at det er et skoleprojekt så kan kravene nemt ændre sig.

Så er der culture, hvilket omhandler hvor godt teamet trives i kaos. Dvs. at det er en kultur hvor der hurtigt kommer ændringer, kommunikationen er vigtig og man skal være omstillings parat hele tiden. Her har vi vurderet, at vi kan godt arbejde i moderat kaos, i og med vi sidder sammen og kun er et team på 5 personer.

Size er hvor stort teamet er, i vores tilfælde, så er vi et team på 5 personer. Derfor tilegner vi os bedst til at arbejde agilt fremfor plandrevet, da det er langt nemmere f.eks. at kommunikerer når man er små teams.

Sidst er der criticality, som handler om hvor store konsekvenser fejl i systemet har. Dvs. om det er ens hjemmelavet app man laver, eller om man laver et vigtigt stykke software til et hospital. I og med det er et skoleprojekt vi laver, har vi derfor vurderet at fejl i vores software ikke har de store betydningsfulde konsekvenser for nogle. Derfor egner det sig også klart bedst til at køre agilt.



Alt i alt vurderer vi at den bedste tilgang til projektet, er at arbejde agilt. Dog da vi bedst kender den plandrevne tilgang, ville det nok være nemmest for os at køre plandrevet. Boehm Turner modellen har altså givet et værktøj til at tage et velbegrundet valgt, samt skabe klarhed i projektplanlægningen.

Metodevalg

Metodevalget i dette projekt bærer præg af konteksten, altså i en undervisnings sammenhæng. Derfor har gruppen, forsøgt at planlægge hele forløbet sådan, at vi får afprøvet så mange af de 13 forskellige primære practices fra XP, samt arbejdet med SCRUM, så vidt det muligt og giver mening. I dette afsnit beskriver vi tankerne bag valgene, samt fordele og ulemper ved tilføjelser eller fravalg, af forskellige practices. Vi forsøgt på forhånd, i Sprint 0, at planlægge projektet og dokumentationen, i de 4 sprints. Hvad der skal laves, hvor starter vi og hvor ender vi, for hvert sprint. Dog med den præmis, at vi arbejder agilt og dermed skal være klar til at omsadle, hvis behovet opstår. For at forklarer metodevalget, vil vi starte med at beskrive vores første iteration af planen, for derefter at kunne beskrive konkrete ændringer, hvad disse skyldes og hvad det gør for resten af projektet.

Overordnet tilvalg for alle sprints

Planen for fordelingen af Accountabilities, er at Scrum Master rollen bliver kørt på turnus. Det for at sikre at alle i gruppen får prøvet denne rolle. Som Product Owner vil vi forsøge at gøre brug af Chat-GPT, vi vil

promte den til at antage den er vores kunde, med et ønske om at få udviklet et distribueret auktionshussystem, med en webpage, et desktopprogram og en database, der virker via et API.

Det gør det muligt for gruppen at oprette User-Stories, der dækker behovene for produktet, i et interview med kunden, og derefter få lavet en prioritering af disse, med acceptkriterier. På den måde har vi sammen med Product owner, fået oprettet en Produkt Backlog. Med de userstories der ligger i product backlog, spiller gruppen planning poker. På den måde bliver UserStories hver for sig vurderet og får storypoints, alt efter hvor omfattende de er. Scrum master agerer i øvrigt også developer under hvert sprint sammen med resten af gruppen.

Det er gruppens plan at følge de fem events i SCRUM. Inden hvert sprint holder vi et sprint planning meeting, hvor vi vælger de UserStories, vi vil arbejde med i sprintet. Vi flytter dem fra product backloggen til sprint backlog. De udvalgte UserStories får tilføjet specifikke tasks, der beskriver hvad der skal laves. Disse får en estimering af timer, på baggrund af UserStoriens storypoints.

Derudover bliver der taget højde for velocity, altså hvad gruppen regner med at kunne nå på et sprint. Da gruppen består af fem og vi har 4 dage i et sprint, arbejder fra 09 til 14, vil vi ca. have 100 produktive timer at gøre med, i hvert sprint. Gruppen regner med at gøre brug af pair programmering og halvere derfor de produktive timer. Vi ender på en estimeret velocity på ca. 50 storypoints pr. Sprint.

For at holde styr på processen vil gruppen, bruge redskabet flying dounut (https://www.flyingdonut.io). Det et online værktøj der gør det muligt at oprette product backlog, sprint backlog og kanbanboard til UserStory/Task management.

Gruppen vil hver dag holde daily scrum. Her gennemgår vi, hvad vi har lavet, hvad vi skal lave på dagen og evt. problemer. Det er de møder der sikre os, at vi overholder tidsplanen, sørger for at kanbanboardet bliver opdateret, så burn down chartet er retvisende.

Det er planen at vi afslutter hvert sprint med et sprint review. Til det deltager vores medstuderende og undervisere. Her er det planen at fremvise funktionel kode og få feedback. Efter sprint review laver vi i gruppen et sprint retrospective. Her går vi sprintet igennem, hvad gik godt, hvad gik ikke godt, er det noget vi ikke nåede og kort ridser op, hvad vi vil have fokus på i næste sprint og eventuelt ændre i tidsplanen.

Overordnet er det gruppens mål at gøre brug af Sit Together når vi arbejder, igennem hele projektet, så vi nemt og hurtigt kan løse problemstillinger og videregive informationer. Generelt er målet at skabe god kommunikation under processen. Det gør det også muligt for gruppen at skabe et Informative Workspace. Vi vil så vidt som muligt have en computer tilkoblet projektoren og derudover gøre brug af whiteboards til informationer, planer osv. til at gøre vores ‘arbejdsplads’ til et Informative Workspace.

Gruppen implementerer, Whole Team, i den forstand at alle koder sammen og derfor kan lære kompetencer til at udvikle systemet og aflevere tilfredsstillende dokumentation på projektet. Når koden skal udvikles, vil vi kode i fællesskab hvis koden er ny eller kompleks. Det er for at sikre vi har den samme forståelse for kritisk funktionalitet. Derefter er det planen at dele gruppen op i mindre grupper og arbejde med Pair-programming. Vi vil arbejde med 2,5 grupper, så vi har to grupper med to og en gruppe med en.

For at sikre at koden stadig kan bygge, på trods af at arbejdet er blevet delt op, er det målet at arbejde med Continuoes Integration, ved ofte at merge branches ned i developement, eller rebase de feature branches vi laver undervejs. Vi planlægger ikke at opsætte automatisk testing under builds, men at køre tests i Visual Studio.

Gruppen benytter også Stories, i form af user-stories. Gruppen vil i tandem med vores simulerede product owner beskrive dem ud fra strukturen:

Hvem gør noget?
Hvad gør de?
Hvad er målet?

Gruppen vil samtidig opstille acceptkriterier, så udviklerne har nogle målbare krav at gå ud fra, når de arbejder på en User Story.

Det også planen at benytte Slack. Gruppen planlægger primært slack opgaver der skal bestå af rapport skrivning, så hvis vi færdiggør de højt prioriterede User Stories i sprintet, vil vi arbejde på dokumentationen af vores proces, samt den teknologiske rapport.

Alt det ovenstående vil indgå i vores weekly cycle, da hvert sprint inkl. sprint review har en længde af præcis en uge.

Tanken bag de fleste af disse valg er som sagt konteksten, altså er hovedmålet at lære under processen, dog med det forbehold at der ikke bliver taget unødige ting med, der eventuelt kan hindre processen. Derfor er

fravalg lige så vigtige som tilvalg. Vi vil forsøge at følge Scrum i forhold til 3-5-3, Accountebilities, Events og Artefakter. Det også gruppens mål at forsøge at forholde sig til Scrum pillars og Scrum Values, det vil vi eksplicit gøre ved at tage stilling til dem ved review/retrospective. Det mener vi, vil give gruppen en god generel forståelse for Scrum og hvordan dette fungere i praksis.

Gruppen har valgt at fokusere på 8 ud af 13 XP-practices, der gennemgående kommer til at være de arbejdsmetodikker der bliver brugt hen over hele forløbet. Vi har taget dette valg på baggrund af erfaring og interesse. Whole Team, Pair Programming, Sit together og Informative workspace er alle nogle practices vi har brugt, uden at vide det, på tidligere projekter og er noget vi alle føler har fungeret godt. De fravalg vi har taget, kan sagtens få en konsekvens, men gruppen har planer om at revurderer processen til hvert sprint retrospective.

Konkret plan for sprints

I sprint 1, vil gruppen hovedsageligt arbejde med Websitet, alt funktionaliteten skal køre med test data, altså stubs. På den måde vil vi hurtigt skabe et færdigt billede af websitet og dermed have noget konkret at tage med til en 'kunde'. Så vil man hurtigt i samarbejde med kunden, kunne danne et billede af slutresultatet. I dette sprint regner vi ikke med at arbejde med nogle nye ting i forhold til scrum og XP og derfor arbejder vi ud fra de otte overordnet ting gruppen har aftalt.

I sprint 2, har gruppen planlagt at arbejde på databasen, altså oprettelsen af DAO klasser, derudover vil gruppen oprette et logiklag i API'et. Ønsket er efter sprint 2 at have det meste backend kodning lavet til hjemmesiden. Så mangler vi kun at binde det hele sammen, så hjemmesiden ikke bruger stubs, men kan testes med testdatabasen. Det også planen i dette sprint at arbejde med TDD i DAO klasserne, vi vil skrive testene til daoklasserne først og derefter skrive koden til at funktionaliteten. Dermed arbejder gruppen med ni Practices fra XP i dette sprint, og stadig følger scrum.

I sprint 3 vil gruppen arbejde på at binde alt back-end logikken sammen med hjemmesiden, derefter vil vi lave en desktop app. Denne skal fungere som et administrativt redskab til hjemmesiden. Her vil vi stadig som udgangspunkt gå ud fra de overordnet ting i forhold til SCRUM og XP, gruppen har aftalt.

Sprint 4, kommer til at være et opsamlings-/afrundingssprint hvor vi gør de ting færdige der evt. måtte være til overs, laver de ting vi vurdere vi mangler og sætter fokus på rapporterne. Det er her vi samler de løse ender, fra de førnævnte sprints. Det er vores buffer sprint, så hvis tidsplanen skrider, kan vi her nå at indhente den. Igen vil vi som i de andre sprints fastholde vores overordnede values og practices fra SCRUM og XP.

Release Plan

Sådan har gruppen planlagt at bruge de fire sprints. Det har vi også formuleret i en release plan for projektet, som kan ses nedenfor ses denne i en tabel.

Sprint	Fokusområder	Centrale aktiviteter	Målepunkt / acceptkriterium
Sprint 1: Top-down web-site	Web-UI + web-server (stub-data)	Udvikle komplet, klikbar webløsning med dummy-data Etablere statiske klasser med stub-bed data til brug i UI. Udarbejde interaktions- og UI-guide sammen med kunden	Kundedemo leverer "look & feel" af det endelige site; alle data-kald stubbet
Sprint 2: DAOLayer & BusinessLogic-Layer	Database-lag + API-logik Test Driven Development	Designe og implementere DAO-klasser samt logik lag. Implementere domænelogik i web-API'et (endpoints) Integrere tests før kode (TDD) for alle DAO-klasser	Test kører grønt UI kan skifte fra stubs til live DB-kald
Sprint 3: Sammenkobling & desktop-klient Minimal viable product	Fuldt flow + desktop-app	Wire alle API-kald ind i websitet Udvikle desktop-administrations-app (Winforms) Udbygge CI/CD-pipeline til to klientplatforme	Web- og desktop-klient trækker live data fra samme API. basis-admin-funktioner virker
Sprint 4: Buffer & rapport	Finish, bug-fixes & dokumentation	Fixe bugs, færdiggøre koden til aflevering Lukke resterende user-stories Skrive rapport	Alle kritiske issues låst Rapport klar til aflevering.

Tabellen her viser, 'Titel', Fokusområde, hvilke centrale aktiviteter samt acceptkriterier for sprintene.

Som man kan læse, er metodevalgene i projektet er præget af, at det er et skoleprojekt, hvor læring er i fokus. Vi har valgt at arbejde ud fra SCRUM-Framework og ni udvalgte XP-practices: Sit Together, Whole Team, Informative Workspace, Pair Programming, Stories, Weekly Cycle, Slack, Continuous Integration og Test-First Programming (TDD).

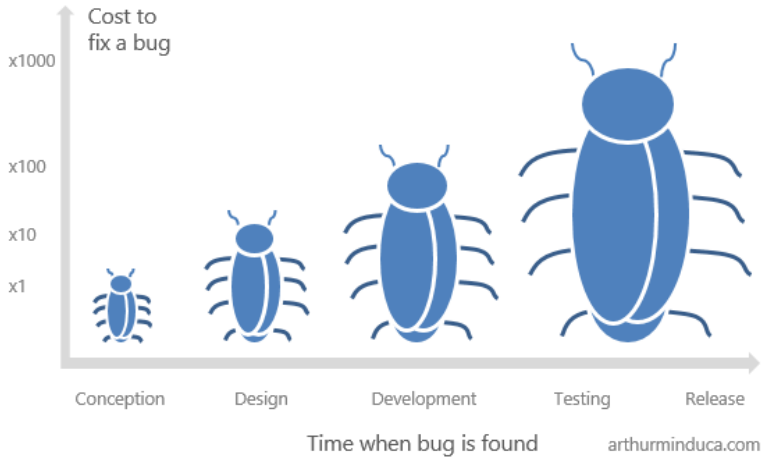
Til gengæld er practices som Ten-Minute Build, Incremental Design, Quarterly Cycle og Energized Work fravalgt. Fravalget skyldes hovedsageligt tidsrammen og fokus på læring frem for fuldt automatiserede processer og avanceret planlægning. Gruppen er dog bevidst om de potentielle konsekvenser, såsom længere feedback loops og manglende langsigtet struktur. Vi forsøger at kompensere med hyppig integration, fælles kodegennemgang og tydelig opgavestyring.

Fordelen ved de valgte practices er, at de styrker vores kommunikation, samarbejde og højner kvalitet, særligt gennem parprogrammering og løbende test. Ulempen ved fravalgene er, at enkelte tekniske forbedringer ikke opnås i samme grad, men disse tages op til evaluering ved projektets afslutning. Overordnet set arbejder gruppen agilt, balanceret og med fokus på at få størst muligt læringsudbytte samtidig med at produktet udvikles systematisk.

Hovedprincipper i planlægning og kvalitetssikring

Inden for kvalitetssikring arbejder man med tre hovedprincipper QM, QA, QC. Quality management er den overordnede filosofi, det er her hvor virksomheden mission og vision bliver afspejlet. Quality assurance er planlægningen af QC med fokus på forebyggelse og tidlig detektering. Quality control er udførelsen af de faktiske kontroller der er udpenslet i QA. [7]

Indenfor quality assurance kan man hyre en QA-Manager til at stå for planlægningen af kvalitetssikringen i et projekt. Denne kan bruge redskaber som DPMV, Risikoanalyse, Praktikker inden for XP osv. Her vil sammenfletningen af alle disse kunne højne kvaliteten i et projekt. Ved at have planlagt det, kan vi minimere bugs eller finde dem tidligt i udviklingsprocessen, billedet under er en visuel repræsentation af skaden ved at finde bug senere i forløbet. [7]



QC handler om udførelsen af QA's planlægning, her vil der blive udført kontrol som f.eks. kodegennemgang, tests og daily standup/daily Scrum. Med denne kontrol gør det muligt at verificere at kvaliteten af produktet. Dette leder dermed videre til en kort beskrivelse af DPMV. D'et betyder at definere, P'et står for planlængningen og M'et for gjort målbar, sidst V'et er handlingen at verificere denne kvalitetssikring. DPM er handlinger inden for QA hvor V er inden for QC. [7]

QM går vi ikke mere i dybden i denne, både her i beskrivelsen af teorien eller bruget af disse i projektet. Dog er det vigtigt at skitsere sammenhængen af disse tre hvilket ses på billedet neden for.

Her ser vi hvordan den ene ikke kan eksistere uden den anden, Først bliver virksomhedens filosofi, vision og mission, dernæst for vi planlagt kvalitetssikringen og slutteligt kontrollere vi disse.

I projektet definerer vi krav som User Stories, prioriterer dem med MoSCoW og gør dem målbare via accept-kriterier. Vores fælles Definition of Done kræver:

- Koden skal kunne bygge
- Alle test består
- Kommentarer i koden
- Nye teknologier og metodikker indføres i rapporten, samt tanker indføres i punktform
- Overholde den kodestandard gruppen har sat
- Flyttes i kanban

I sprint 2 vil vi gøre brug af TDD, her er der derfor fokus på at sikre acceptkravene ved hjælp af test baseret på disse. Derefter implementeringen af koden så alle testene består, dette vil helt klart være med til at sikre både kodestandarden og kvaliteten. Derudover er implementeringen af diverse XP Pracitices som, Pair Programming, Whole Team, Continuos integration, Daily Standup og Weekly Cycle med til at højne og sikre kvaliteten af projektet i hver deres form. Det store fokus på feedback inden for agile arbejdsmetodikker, hjælper også kvaliteten.

Risk management

Med risk management er det muligt at få et overblik, samt styr på alt hvad der kan gå galt og hvordan vi skal håndtere det. I tabellen under ser vi vores risiko analyse, den er udformet ved hjælp af IAPM, der står for; Identify, Analyze, Plan, Monitor[8]. I denne model har vi skrevet de 12 risici ned som er mest relevant ift. denne opgave. Først har vi identificeret dem, for derefter at give dem point, i henholdsvis chance og effekt. Derefter rangeret dem efter flest point og så har vi lavet et cut off ved 12 point. Dette betyder alt op til og med 12 bliver irrelevant da det har en så lille effekt eller chance. Efter vi har rangeret, har vi lavet en plan for håndtering af disse ricisi.

Prioritering	Risiko	Chance	Effekt	total	Plan for håndtering
1	Tidsplan bliver ikke overholdt	6	10	60	- Løbende sprint-planlægning med klare mål og timebox. - Daily SCRUM og statusopdateringer. - Synlig og prioriteret product backlog med mulighed for at skære fra, hvis der opstår forsinkelser. - Buffer-tid i planen
2	Krav til opgaven bliver ikke opfyldt	4	10	40	-Løbende møder med vejleder, med henblik på at sikre god og korrekt opgaveløsning -Rådfør studentervejledningen, for konkrete krav til opgaven
3	Fravær og sygdom i gruppen	10	3	30	-Par programmering og deling viden -Dokumentation af kernefunktioner -Kommuniker dagligt på Discord

4	Mangel af kompetencer i gruppen	5	5	25	-Spikes til vigtige emner -parprogrammering for vidensdeling -løbende møder med vejleder for at sikre vi er på rette spor
5	Uklare acceptkrav til user stories	4	6	24	Test driven developement Klare Definitions of done
6	Geopolitisk påvirkning	2	10	20	Vi retter os efter statens udmeldinger, bliver meget religiøse og glemmer alt om skole.
7	Valg af forkert arkitektur til systemet	2	10	20	Vi holder os til den udleverede arkitektur.
8	Samarbejdsproblemer i gruppen	2	6	12	Cut Off
9	Databasen (som gruppen benytter) går ned	1	10	10	Cut Off
10	Hardware nedbrud på en eller flere maskiner i gruppen	2	3	6	Cut Off
11	WiseFlow har nedbrud på et kritisk tidspunkt	1	5	5	Cut Off
12	GitHub gå ned	1	3	3	Cut Off

Ved at vi har lavet en risiko plan før projektet gik i gang, vil det kunne hjælpe os med at fange problemer før de opstår.

Kvalitetskriterier og arkitektur

Kvalitetskriterier

Vi har nogle krav som er gældende for hele systemet som vi bruger til kvalitetssikring. De ser ud som følgende:

Sikkerhed

- Krypter følsomme data.
- Gem system- og virksomhedsdata i en sikret database.
- Giv kun adgang til autoriserede, autentificerede brugere.
- Beskyt API'et med en nøgle.

Usability

- Fuld support til moderne browsere, mobiler og OS'er.
- GUI skal være intuitiv og brugervenlig.

Reliability

- Bevar historik; undgå hard delete.
- Minimer nedetid og tag regelmæssige backups.
- Udrul opdateringer i staging for næsten nul produktion-nedetid.

Performance

- Hold responstid lav med caching.
- Skalér til mange samtidige brugere uden performance-tab.
- Designet skal være horisontalt skalérbart.

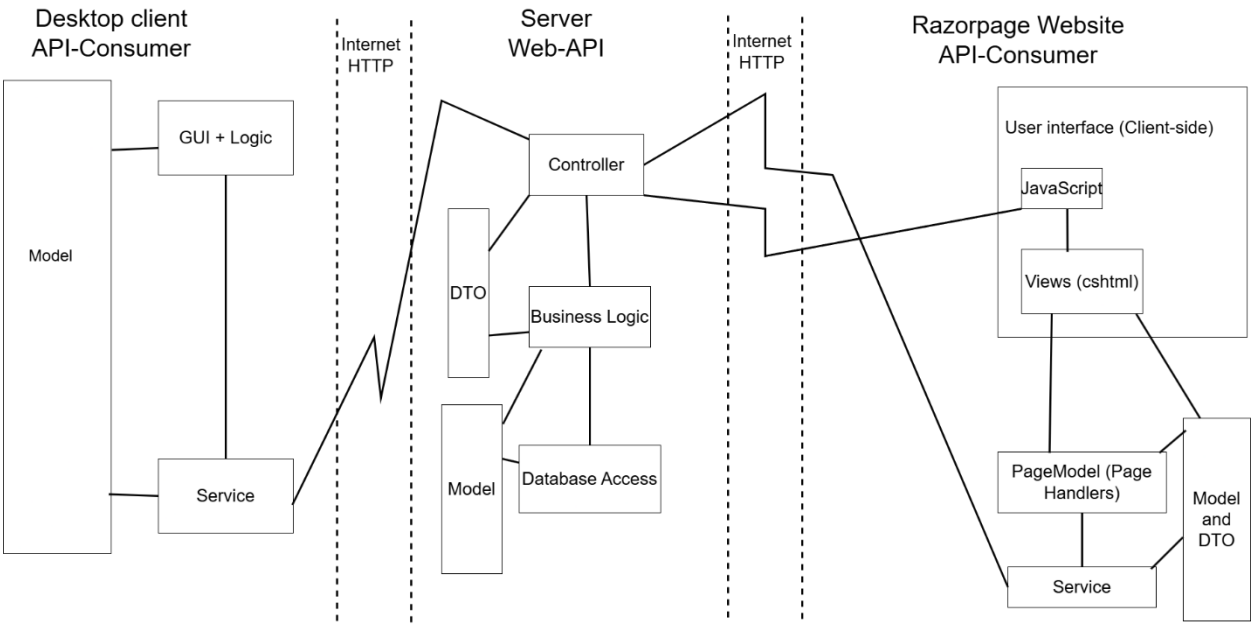
Supportability

- UI og dokumentation skal være på engelsk.

Disse krav har vi altid i baghovedet, og de er med til at forme de valg vi tager om at udvikle systemet. Sikkerhedskravene hjælper os med at udbygge vores adgangskontrol og integritetssikring af systemet.

Usability og Supportability kravene sætter fokus på brugeroplevelsen og guider vores frontend udvikling. Reliability kravene er med til at sikre dataintegritet og availability. Performance kravene er sammen med usability kravene med til at sikre den gode brugeroplevelse.

Arkitektur



I den strenge definition af "tier" vil, vores arkitektur være et 3-tier system der består af et ASP.NET webapi og en webapp (hjemmeside) i form af en razorpages app. Derudover vil der være en desktop klient som også kan interagere med API'et. Disse 3 komponenter vil køre på 3 forskellige maskiner, hvilket vil gøre til et 3-tier system. Hvis vi udbreder definitionen af, "tier" til en klient-server-sammenhæng hvor en tier primært er defineret af sin rolle som enten en klient, der anmoder om tjenester, eller en server, der leverer tjenester. Vil vores system være et 2-tier system. Da både desktop app'en og websitet er klienter, som anmoder om tjenester fra vores server, nemlig web api'et.

I forhold til lag på de forskellige tiers ser det ud som følgende: API'et vil bestå af 3 lag, controller, logik og data access laget. Hjemmesiden skal bestå af 3 lag: frontend (client side html, css, JS), pagemodel laget, som står for serverside rendering og logik, og et service lag som snakker med API'et.

Kriterierne sammen med arkitekturen skal være med til at danne grundlaget for vores beslutninger under udviklingen. Kriterierne sikrer at vores system lever op til de krav om brugervenlighed, sikkerhed, stabilitet og performance vi har stillet. Arkitekturen og lagopdelingen skal være med til at øge fleksibiliteten og skalerbarheden af vores kode.

Refleksion

Formålet med dette afsnit er at reflektere over de erfaringer, gruppen har gjort sig under forløbet og dele dem med læseren. De præsenteres kronologisk, i tråd med gruppens arbejdsgang. Vi starter derfor med Sprint 0.

Sprint 0 bar tydeligt præg af, at gruppen ikke tidligere havde arbejdet med agile metoder. Meget af arbejdet blev revurderet eller skrottet. Et eksempel er vores product backlog, som oprindeligt bestod af seks userstories.

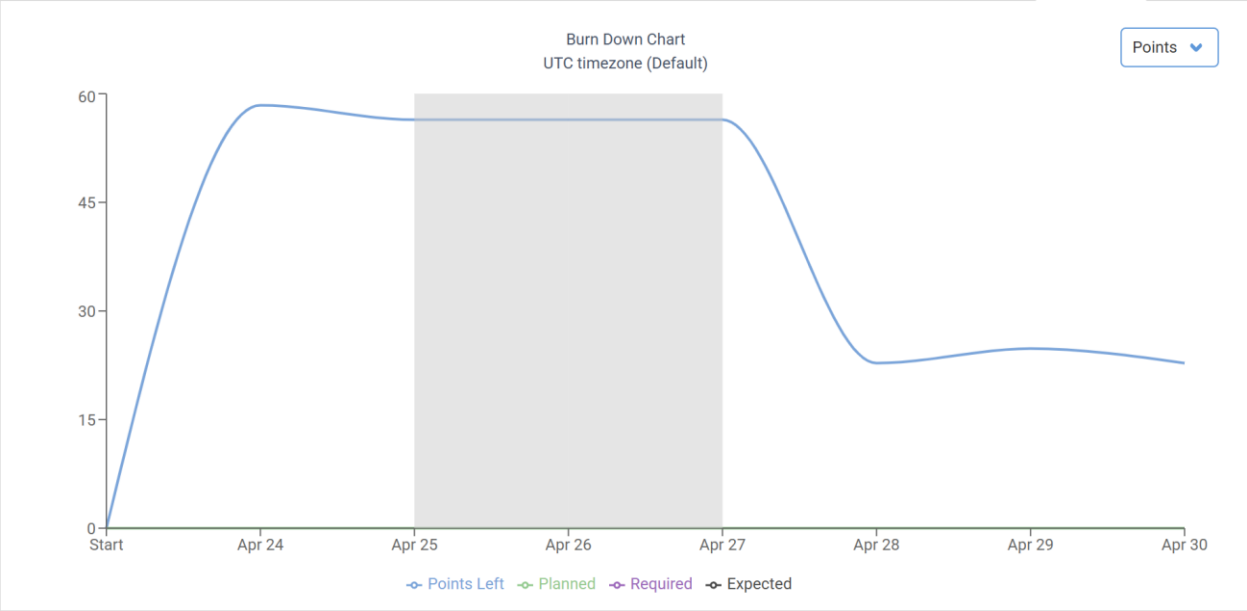
Vi valgte at opdele kodningen og sprintene efter vores tiers, og derfor fungerede de oprindelige userstories ikke. Vi ændrede dem til ca. 30, opdelt i frontend og backend, men mange blev for detaljerede og overlappede, og blev dermed mere til tasks, hvilket ikke stemmer overens med, hvordan man bør arbejde med userstories i Scrum.

Backloggen blev estimeret via Planning Poker, men da vi antog, at ét storypoint svarede til én time, blev estimaterne misvisende, og det gav udfordringer senere i processen. Efter hvert sprint har vi afholdt retrospectives, hvor vi har gennemgået, hvad der gik godt og hvad der skulle forbedres. De næste afsnit bygger derfor på refleksioner fra disse møder.

I Sprint 1 erfarede vi hurtigt, at userstories ikke fungerede særligt godt, mange overlappede og var brudt helt ned til task-niveau. Det gjorde, at man kunne have løst flere "userstories" på én gang, hvilket skabte rod og gjorde, at gruppen mistede overblikket over, hvad der var lavet. Fokus på rollerne mistede vi undervejs, og derfor blev DOD ikke overholdt.

De ting, gruppen erfarede gik godt, var vores branching-strategi. De otte XP-principper blev overholdt og fungerede rigtig godt. Daily Scrum blev udført hver dag og var en af de ting, der gjorde, at overblikket ikke blev helt tabt i processen. Derudover valgte vi at lave Sprint Planning Meeting inden sprintets start, hvilket betød, at vi på dag ét kunne gå direkte i gang med kodning.

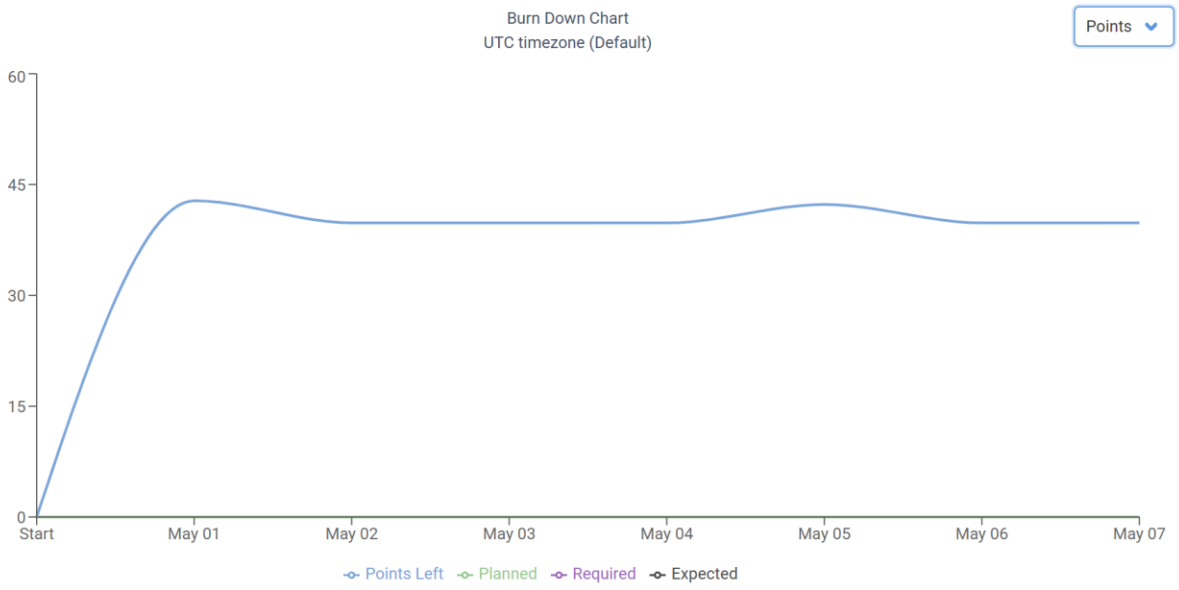
Med disse erfaringer satte gruppen sig ned og blev enige om, hvad der skulle gøres bedre til Sprint 2. Her aftalte vi et større fokus på roller, hvor Scrum Masteren skulle lægge stor vægt på, at den generelle DOD blev overholdt. Vi ville revurdere userstories, så de ikke overlappede. Vi ønskede at fortsætte med XP-principperne og bevare den positive tilgang til processen.



I Sprint 2 erfarede vi, at vores tilgang til Kanban-boardet og Flying Donut ikke var optimal. Vi havde forsøgt at revurdere de userstories, vi tog med ind fra product backloggen, men problemet var, at vi helt skrottede userstory-strukturen og mere beskrev i praksis, hvad vi skulle lave, og lavede tasks ud fra det. Denne tilgang viste sig hurtigt ikke at være optimal, og vi mistede igen overblikket over Kanban-boardet, hvilket vores burndown chart tydeligt viser.

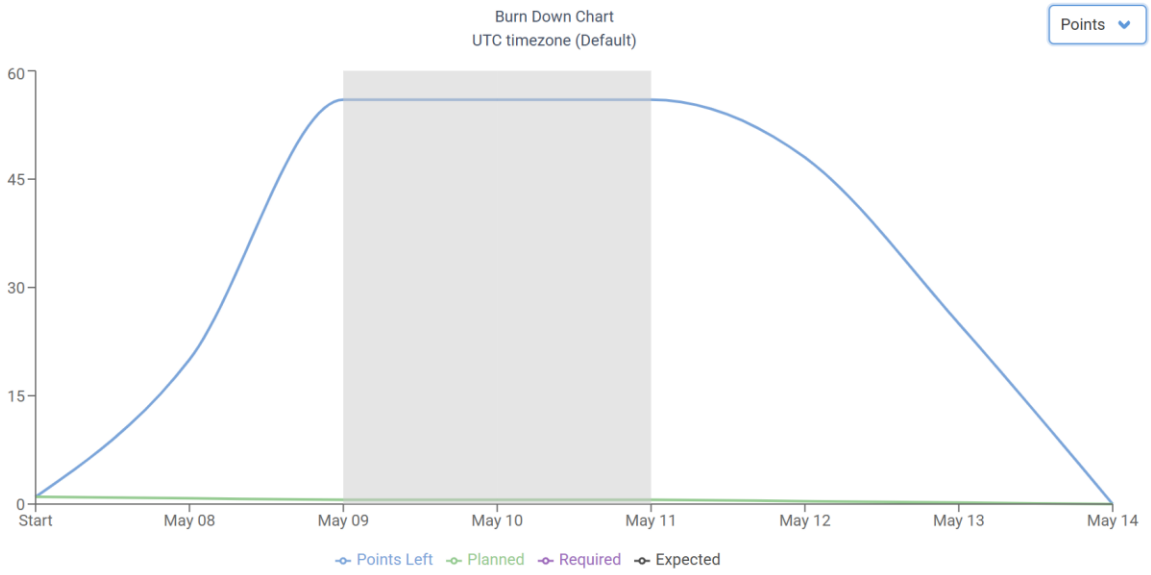
På dette sprint havde vi yderligere fokus på TDD, hvilket fungerede godt. Med dette nye tiltag var det vigtigt, at alle forstod metoden, hvilket gruppen løste godt. Gruppen erfarede dog, at det ikke er en arbejdsmetodik, vi vil arbejde videre med, da arbejdsprocessen var drøj og meget tidskrævende. Vi har dog formået at overholde den gode arkitektur, gruppen på forhånd havde planlagt.

Tingene, der skulle gøres bedre til Sprint 3, var endnu større fokus på roller og endnu engang en revurdering af userstories, da den nye implementering heller ikke fungerede ideelt.

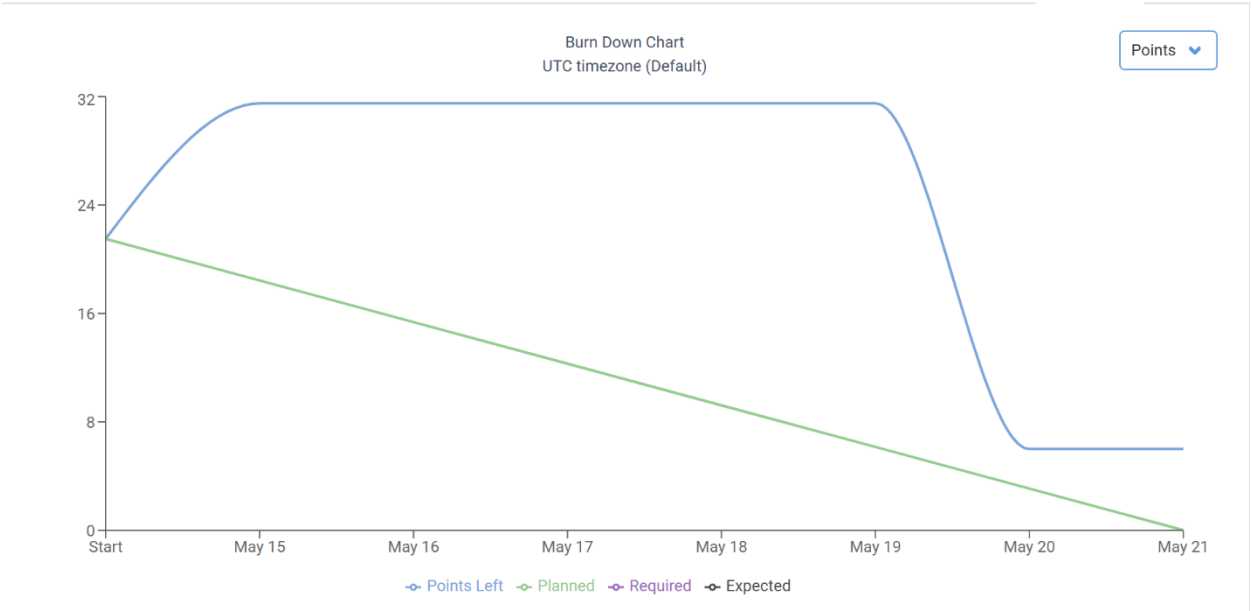


I Sprint 3 erfarede vi igen, at gruppen ikke overholdt DOD. Derudover brugte gruppen en del tid på refactoring af kode på grund af quickfixes i tidligere sprints. Gruppen valgte efter vejledning at skifte fokus til minimal viable product, altså at håndtere samtidighedsproblemet og have distribueret system samt desktop-app klar ved sprintets afslutning. Dette gjorde også, at brugen af userstories fungerede langt bedre i dette sprint. Ændringen kan også ses på burndown chartet, hvor vi nu når til bunds i alle vores userstories, dog bærer chartet præg af, at ændringen først skete torsdag eftermiddag, og derfor går kurven opad i starten.

Set i lyset af, at vi havde en minimal viable product efter dette sprint, var der ikke meget andet tilbage end at færdiggøre rapporten i Sprint 4 og derefter finpudse produktet. Vi mener, at Sprint 3, efter et par ugers arbejde med de agile arbejdsmetoder og styringsværktøjer, var det sprint, hvor det fungerede bedst for gruppen. Det resulterede også i det flotteste burndown chart til dato.



Sprint 4 blev brugt præcis som planlagt, vi har finpudset produktet og skrevet rapport. Burndown chart i denne periode viser tydeligt, at rapportskrivning har fyldt meget og ikke var opdelt i tasks, hvilket medførte et kolossalt dyk i charten, da rapporten blev færdig.



Samlet set over samtlige sprints har arbejdet med Scrum og XP fungeret rigtig godt. Vi har fulgt de events og benyttet de artefakter, Scrum indeholder. Arbejdet med roller har dog været kringlet. Det har været svært at arbejde struktureret med på grund af de meget flydende roller i dette projekt. Scrum-master har også været developer, og Product Owner var i starten AI, men gik senere på runde mellem gruppemedlemmer.

Som nævnt har gruppen haft problemer med DOD, fordi den ikke altid blev fulgt. Grunden til dette var, at rapportskrivning var en del af DOD, men også blev betragtet som en slack-opgave. DOD skal jo være opfyldt efter hver userstory, hvor slack-opgaver netop kan udskydes, hvilket ikke harmonerer. Gruppen burde derfor enten have fjernet rapportskrivning fra DOD eller beholdt den og ikke haft den som slack-opgave.

Generelt har XP-principper, som vi tidligere brugte uden at vide, at det var praksisser, fungeret rigtig godt. De har givet højere kodekvalitet, instant code review, fælles forståelse for systemet og nye teknologier. Processen har givet os teoretisk begrundelse for noget, vi i praksis selv var kommet frem til.

De nye principper som Weekly Cycle, Continuous Integration og TDD har givet teoretisk forståelse og styrket de nævnte punkter. TDD har på sin vis erstattet vores tidligere brug af kommunikationsdiagram, fordi det tvinger os til at tage stilling til metoder, navngivning og placering af logik. Gruppen erfarede dog, at denne praksis er meget tidskrævende og sandsynligvis mere effektiv, hvis man har mere tid og/eller erfaring.

En vigtig læring er Incremental Design. I bagklogskabens lys burde vi have implementeret denne praksis – vi lavede for tidligt en kompleks struktur og inddrog domæneregler, fremfor at fokusere på core-userstories og senere bygge videre.

Overordnet set har vi været glade for den agile arbejdsmetodik og set dette som en meget lærerig proces.

Konklusion

Et distribueret realtidsauktionshussystem, kan udvikles både plandrevet og agilt. Da projektet ikke er en proces hvor der skal udvikles en innovativ løsning til et problem, men et produkt som allerede eksisterer, hvor funktionaliteten er kendt og udviklet, taler teorien egentligt for at plandreven ville være en fornuftig arbejdsmetodik.

Vores boehm og turner model indikerer at vi bør arbejde agilt, da samtlige vurderings kriterier, med undtagelse af erfaring i gruppen, taler for en agil arbejdsmetodik. Kulturen i gruppen er også en stor fortaler i brugen af agil udvikling. Gruppen har nu arbejdet sammen siden første semester, vi har i praksis brugt flere af XP-practices i tidligere semestre og det taler for at gruppen arbejder bedst agilt.

Efter projektforløbet har vi konstateret at vi, som de fleste moderne virksomheder, højest sandsynligt ville vælge en scrum-but tilgang, hvor vi vælger de elementer fra XP og scrum der passer til vores gruppe, og fravælger dem vi syntes ikke giver mening for projektet.

Litteraturliste

[1] Ken Schwaber og Jeff Sutherland, "Scrum Guide". Set: 22. maj 2025. [Online]. Tilgængelig hos: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Danish.pdf

[2] Kent Beck og Cynthia Andres, *Extreme Programming Explained: Embrace Change.*, Kindle Edition. Pearson Education, 2003.

[3] "Hvad er Extreme Programming (eller XP)?" Set: 22. maj 2025. [Online]. Tilgængelig hos: https://blivprojektleder.dk/extreme-programming/

[4] Jakob Farian Krarup, "Introduction to eXtreme Programming". Set: 22. maj 2025. [Online]. Tilgængelig hos: https://ucndk.sharepoint.com/:p:/r/sites/Section_0210007/_layouts/15/Doc2.aspx?action=edit&sourcedoc=%7B54dde8f5-6869-46c1-afda-63460ced712c%7D&wdOrigin=TEAMS-MAGLEV.teamsSdk_ns.rwc&wdExp=TEAMS-TREATMENT&wdhostclicktime=1747381063298&web=1

[5] Jakob Farian Krarup, "Kanban". Set: 22. maj 2025. [Online]. Tilgængelig hos: https://ucndk.sharepoint.com/:p:/r/sites/Section_0210007/_layouts/15/Doc2.aspx?action=edit&sourcedoc=%7B2ae121b3-3d01-439e-93d3-2b3a291acb63%7D&wdOrigin=TEAMS-MAGLEV.teamsSdk_ns.rwc&wdExp=TEAMS-TREATMENT&wdhostclicktime=1747910267423&web=1

[6] UVA CS 3240, "The Polar Chart". Set: 22. maj 2025. [Online]. Tilgængelig hos: https://www.youtube.com/watch?v=Kl21VvULhj8

[7] Jakob Farian Krarup, "Quality Assurance". Set: 22. maj 2025. [Online]. Tilgængelig hos: https://ucndk.sharepoint.com/:p:/r/sites/Section_0210007/_layouts/15/Doc2.aspx?action=edit&sourcedoc=%7B75ae06a4-e785-4504-bda9-5017d350e539%7D&wdOrigin=TEAMS-MAGLEV.teamsSdk_ns.rwc&wdExp=TEAMS-TREATMENT&wdhostclicktime=1747910749939&web=1

[8] Jakob Farian Krarup, "Risk Analysis and Management". Set: 22. maj 2025. [Online]. Tilgængelig hos: https://ucndk.sharepoint.com/:p:/r/sites/Section_0210007/_layouts/15/Doc2.aspx?action=edit&sourcedoc=%7B58324968-f26f-4353-a665-4e31aa079fd0%7D&wdOrigin=TEAMS-MAGLEV.teamsSdk_ns.rwc&wdExp=TEAMS-TREATMENT&wdhostclicktime=1747911047860&web=1

Bilag

Bilag 1

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation
2	Able to tailor a method to fit a precedented new situation
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience can become Level 2.
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.