HÖGSKOLAN
DALARNA

Examensarbete

Kandidatnivå

Clean coding i team

En fallstudie om hur ett team går tillväga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer

Clean code in team - A case study to describe how a team works to establish a common mindset based in the guidelines of Clean code

Författare: Emelie Emretsson
Handledare: Anders Avdic
Examinator: Pär Eriksson
Ämne/huvudområde: Informatik
Kurskod: IK2017
Poäng: 15 hp
Ventilerings-/examinationsdatum: 1 juni 2017

Sammanfattning

Idag byggs många system som består av svårlästa kodbaser med låg förvaltningsbarhet. En anledning till detta är att utvecklarna av systemet har olika bakgrund och kunskap i hur de skriver kod. Att skriva sin kod på helt skilda sätt är något som kan skapa problem i takt med att system blir större och allt mer komplexa.

Nethouse i Borlänge har sedan 2015 arbetat med förvaltningsuppdraget TRAP (Transportstyrelsens Administrativa Processystem) där en problematiskt förvaltning upplevts i och med att systemet är uppbyggt med hjälp av olika tekniker. Tekniken i TRAP ska lyftas och målet med detta är att skapa en mer lättläst och förvaltningsbar kodbas jämfört med hur TRAP ser ut idag. För att uppnå detta är planen att i teamet etablera ett gemensamt tankesätt som grundas i de riktlinjer som Clean code förespråkar.

Studien syftar till att beskriva hur ett team arbetar med etableringen av ett gemensamt tankesätt som grundas i Clean Codes riktlinjer samt faktorer som anses vara viktiga att beakta.

För att uppnå syftet användes två frågeställningar:

- Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?
- Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

En fallstudie utfördes med intervjuer och enkäter som datainsamlingsmetoder för att ha möjlighet att besvara frågeställningen.

Resultatet från studien visar att teamet på Nethouse använder sig av par- och mobprogrammering samt i enstaka fall kodgranskning för att etablera det gemensamma tankesättet. Resultatet beskriver även fyra faktorer som är viktiga att beakta när ett gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras. De fyra faktorerna är ömsesidigt förtroende, ömsesidighet kring det arbete som utförs, tvåvägskommunikation och tillvägagångssätt.

Nyckelord: Clean code, gemensamt tankesätt, riktlinjer, team, betydande faktorer, big five in teamwork

Abstract

Many of todays systems are made of code bases with low readability which leads to low maintainability. One reason to this is that developers of the system have different experience and knowledge in how to write code. When code is written in totally different ways it can create problems as the system grows and becomes more complex.

Since 2015, Nethouse in Borlänge has managed a system called TRAP (Transportstyrelsens Administrative Process System). TRAP is built with different techniques and during the maintainability process a lot of problems has occured because of that. The technique in TRAP is about to be lifted and by doing this one part of the goal is to create a code base which is more easy to read and maintain compared to todays code base. To achieve this goal the plan is to establish a common mindset in the team. A common mindset which is based in a set of guidelines called Clean code.

The purpose of this study is to describe how a team is working to establish a common mindset based in the guidelines of Clean code and to describe important factors to consider in this situation.

Two research questions was used to achieve the purpose of this study:

- How is the team working today to establish a common mindset?
- Which factors can be considered as important when a common mindset is about to establish?

A case study with the help of interviews and questionnaries was conducted to answer these two questions.

The result shows that the team is using pair programming, mob programming and also code review to establish a common mindset. The result also shows that the four factors mutual trust, mutual performance monitoring, closed loop communication and method are more important to consider in this situation.

Keywords: Clean code, common mindset, guidelines, team, important factors, big five in teamwork

Förord

Uppsatsen är resultatet av det examensarbete som utförts på det systemvetenskapliga programmet vid Högskolan Dalarna under våren 2017.

Jag vill tacka min samarbetspartner Nethouse och mina två handledare på företaget, Sofia Blomgren och Jens Malm för att ni ställt upp och hjälpt mig under arbetets gång. Jag vill även tacka övriga delar av TRAP-teamet som ställt upp genom att delta i intervjuer och enkäter.

Jag vill även tacka min handledare vid Högskolan Dalarna, Anders Avdic som väglett mig under arbetets gång.

Emelie Emretsson
24 maj 2017

Innehållsförteckning

Figurförteckning

Tabellförteckning

1. Inledning

I detta kapitel beskrivs en bakgrund (1.1) till studien. En problemformulering med tillhörande problemfrågor (1.2) presenteras samt även studiens syfte (1.3) och avgränsning (1.4).

1.1 Bakgrund

Ett IT-system hos en verksamhet ses idag som en självklarhet för de flesta. Utvecklingen av nya IT-system påbörjas dagligen och många oroar sig över de höga kostnader som utveckling och införande av ett nytt IT-system medför. (Srinivasulu, 2012)

Utvecklingsfasen är en av de kortare faser som ingår i ett systems livscykel. I utvecklingsfasen skapas en grund inför förvaltningsfasen som är den längsta i ett systems liv. Förvaltningsfasen sträcker sig från att systemet implementeras till att det avvecklas och enligt Haverblad (2009) uppstår 70-80 % av systemets totala kostnad under förvaltningsfasen. Programmerare kan redan i utvecklingsfasen tänka på framtida förvaltning genom att lägga vikten på välskriven kod vilket således kan hjälpa till att öka förvaltningsbarheten, som i sin tur minskar förvaltningskostnaderna. (Martin, 2009) Förvaltningsbarhetsindex kallas det kvalitativa mått som kan användas för att mäta förvaltningsbarheten av ett system. (Heitlager, Kuipers & Visser, 2007) Förvaltningsbarhet innebär dock inte endast kvalitativa mått utan även andra aspekter. Dessa aspekter kan vara hur lång tid det tar att lokalisera vart problem i koden uppstått och möjligheten att ändra en del av kodbasen utan att det påverkar för mycket i en annan. (Rosene et al., 1981)

Figur 1. Livscykeln som beskriver ett systems liv. (Systems development life cycle, 21 maj, 2017)

Det finns olika faktorer som påverkar att det idag byggs komplexa system som består av oläsliga och svårförvaltade kodbaser. Två av dessa faktorer är tidspress samt programmerares olika sätt att skriva sin kod på. (Lerthathairat & Prompoon, 2011) Tidspress är en påverkande faktor som kan vara svår att kontrollera. Däremot hur den individuella programmeraren skriver sin kod går å andra sidan att förändra.

> *”The only way to make the deadline – the only way to go fast – is to keep the code as clean as possible at all times.”* – Martin (2009) (s. 30)

Robert Cecil Martin, även kallad Uncle Bob, är grundaren till begreppet Clean Code. Det är en uppsättning riktlinjer för hur en programmerare bör skriva sin kod för att den ska bli så ren som möjligt. Med uttrycket ren så menas exempelvis att en metod endast ska utföra en uppgift och samtidigt vara relativt kort. (Se mer under kap 2.2) (Martin, 2009)

Nethouse är ett IT-företag som finns på flera orter i Sverige, bland annat i Borlänge. De tre huvudområden som Nethouse är verksamma inom är affärs- och verksamhetsutveckling, IT-infrastruktur samt systemutveckling och förvaltning. På kontoret i Borlänge så är man verksam inom affärs- och verksamhetsutveckling samt utveckling och förvaltning.

I dagsläget utförs oftast systemutveckling och systemförvaltning i team, precis som hos Nethouse i Borlänge. (Dingsøyr & Dybå, 2012) Ett team definieras av Dyer (1984) och Salas et. al (1992) som två eller fler individer som interagerar och dynamiskt arbetar tillsammans mot ett gemensamt uppsatt mål. I jämförelse med en enskild individ har ett team större potential att anpassa sig, vara produktiva och även generera mer kreativitet. Den delade uppfattningen av det gemesamma målet tillsammans med de

uppgifter som behöver göras för att nå målet kan definieras som mentala modeller inom teamet. (Salas, Sims & Burke, 2005)

Sedan 2015 har teamet på Nethouse arbetat med att förvalta ett större IT-system åt Transportstyrelsen (TS). Detta system består för närvarande utav 11 olika delsystem och benämns TRAP, vilket är en förkortning av Transportstyrelsens Administrativa Processystem. TRAP används utav handläggare på Transportstyrelsen för att hantera ärenden inom järnvägssektorn. Systemet används som hjälpmedel för att exempelvis granska och godkänna lokförarbevis, granska och bevilja fordonstillstånd som är verksamma inom järnvägen och även registrera och bedöma händelser som sker på och kring järnvägen. TRAP har utvecklats succesivt sedan starten år 2007 vilket medfört att de 11 delsystemen är uppbyggda med hjälp av olika tekniker så som ASP.NET webforms och ASP.NET MVC. Under åren har många olika utvecklare deltagit i utvecklings- och förvaltningsarbetet vilket påverkat kodstandarden i systemet.

I och med att delsystemen skiljer sig åt teknikmässigt så påverkar detta förvaltningen då en liten förändring kan ta mycket längre tid än planerat och det blir svårt att avgöra hur mycket arbete som krävs vid en förändring. Driftmiljön som delsystemen driftas i börjar bli föråldrad vilket innebär att den behöver uppgraderas. För att lyckas med detta så måste tekniken i vissa delar av delsystemen lyftas över till nyare teknikval.

Det nuvarande TRAP ska förnyas och Nethouse har nyligen påbörjat ett utredningsarbete för att titta närmare på hur tekniken i systemet kan lyftas vilket kommer innebära att kod kommer refaktoriseras och återanvändas i den nya tekniken. Med refaktorisering menas att programkod skrivs om för att exempelvis förbättra läsbarheten. Refaktorisering innebär att funktionaliteten utåt sett inte förändras. (Fowler & Beck, 1999) Både förvaltnings- och utredningsarbete ingår i förvaltningsuppdraget TRAP.

Tanken är att de 11 delsystemen i framtiden ska slås samman till ett gemensamt system. Ett av målen med detta arbete är att skapa en mer lättläst och förvaltningsbar kodbas jämfört med hur TRAP ser ut idag. För att uppnå detta är planen att i teamet etablera ett gemensamt tankesätt som utgår från de riktlinjer som Clean code förespråkar.

1.2 Problemformulering

Sedan Nethouse tog över förvaltningsuppdraget TRAP så har förvaltningen många gånger upplevts som problematisk då nuvarande lösning är utvecklad med hjälp av varierande tekniker. Förvaltningsbarheten har ansetts som låg i och med att förändringar i koden på ett ställe i sin tur orsakat större problem i andra delar av koden. Den tidigare problematiska förvaltningen gör att man under förvaltningen av nuvarande TRAP och framförallt inför införandet utav ny teknik vill tillämpa ett uttalat gemensamt tankesätt, ett tankesätt som kommer grunda sig i de riktlinjer som Clean code förespråkar. Målet är att skapa en bra förutsättning för framtida förvaltning av TRAP genom att vid övergången till ny teknik skriva en lättläst och förvaltningsbar kodbas. En fråga som uppstår ur denna situation är:

Hur går man som team till väga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer?

För att kunna svara på frågan har två underfrågor valt att användas:

- Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?
- Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

1.3 Syfte

Syftet med studien är att beskriva hur ett team går tillväga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer. Studien syftar även till att beskriva betydande faktorer som anses vara viktiga att beakta när ett gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras.

1.4 Avgränsning

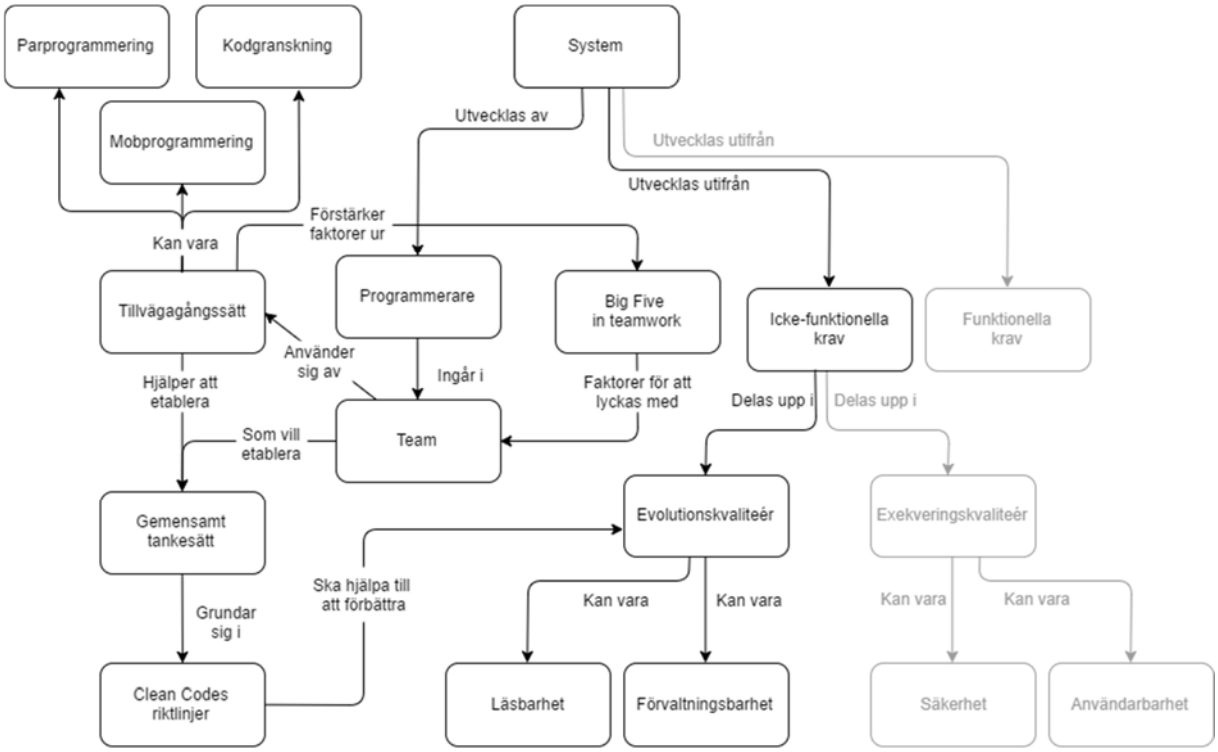
Studien som utförs är en fallstudie vilken syftar till att endast studera teamet hos Nethouse i Borlänge och hur de jobbar med etableringen av det nya gemensamma tankesättet som grundas i Clean codes riktlinjer. Studien är avgränsad till att endast undersöka hur teamet går till väga för att etablera detta gemensamma tankesätt. Den syftar inte till att undersöka hur det gemensamma tankesättet påverkar läsbarhet och förvaltningsbarhet eftersom utvecklingen av nya TRAP och även etableringen av det gemensamma tankesättet pågår under en längre process än tiden då studien genomförs.

2. Teori

I detta kapitel beskrivs centrala begrepp och ramverk mer ingående som är av betydelse för studien.

2.1 Begreppsgraf

Nedan har en begreppsgraf tagits fram (se figur 2) för att visa hur de olika begreppen som tas upp i teoriasvnittet hör ihop med varandra.



Figur 2. Begreppsgraf över hur de olika begreppen i teorin hänger ihop med varandra.

2.2 Clean Code

Det finns egentligen ingen tydlig definition av vad Clean code är. Grundaren till begreppet, författaren Robert C. Martin även kallad Uncle Bob, väljer att inte sätta en självklar definition på begreppet. Han har i sin bok ”*Clean code: a handbook of agile software craftsmanship*” samlat en handfull personer som på något vis haft en betydande roll för programmeringen. Dessa personer har fått lämna sin egen definition av begreppet Clean Code och nedan följer två av dem. (Martin, 2009)

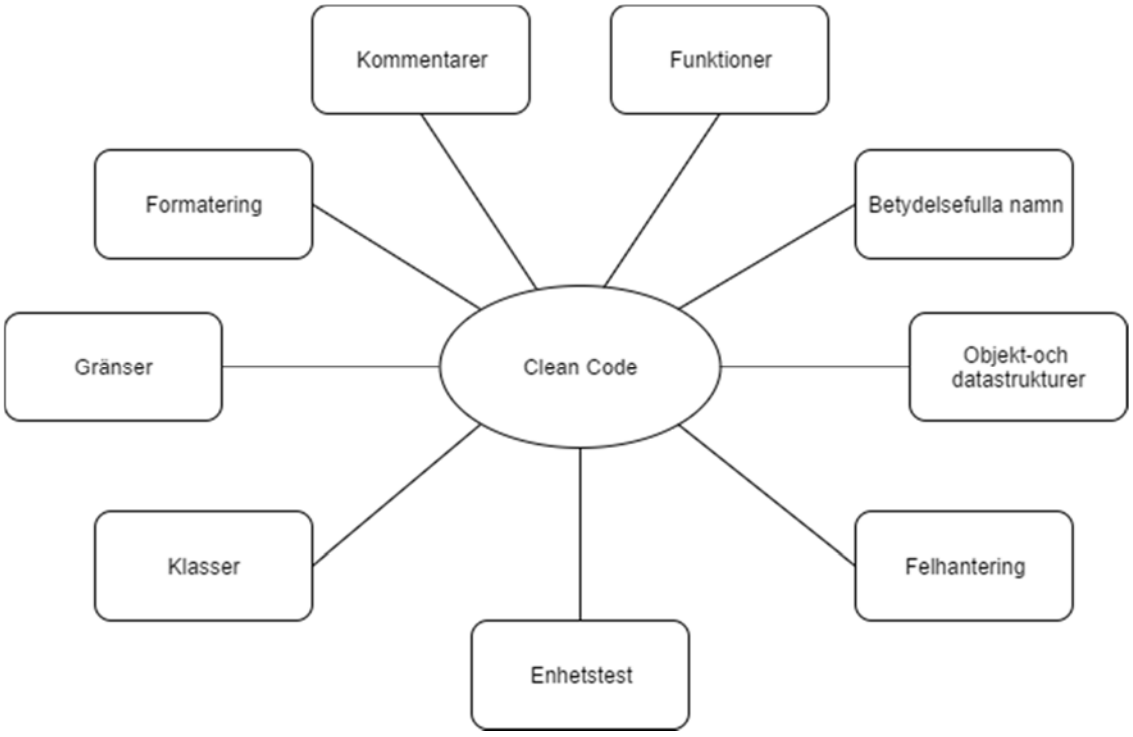
Den första definitionen i Martins (2009) bok ges av Bjarne Stroustrup, grundaren och författaren av programmeringsspråket C++. Bjarne definierar Clean code som att logiken i koden ska vara rakt på sak, vilket minskar risken för buggar att gömma sig. Han nämner även att beroenden som uppstår i koden minskas vilket bidrar till lättare förvaltning. (Martin, 2009)

En annan definition av begreppet ges av Ward Cunningham, en amerikansk programmerare vilken grundade det såkallade wikikonceptet. (Ward Cunningham, 2016, 23 oktober) Wards definition av Clean Code handlar om att när man som programmerare jobbar med bra kod så märks det tydligt.

> *”You know you are working on clean code when each routing you read turns out to be pretty much what you expected”* – Ward Cunningham, (Martin, 2009) (s. 35)

När all kod som man läser gör det den ska och inte andra oförutspådda saker, då är det Clean code. Ward menar att det går att kalla kod för vacker kod när den får programmeringsspråket att se ut att vara skapat för att lösa problemet. (Martin, 2009)

Vad är det som gör att denna kod, som kallas Clean code, skapas? Martin (2009) beskriver i sin bok ett antal riktlinjer som en programmerare kan ta hjälp av för att lära sig skriva Clean code. De uttalade riktlinjer som nämns av Martin (2009) har sammanställts i figuren nedan för att ge en tydligare överblick. (Se figur 3)



Figur 3. Riktlinjer som ingår i begreppet Clean code. (Martin, 2009)

Sammanställt så förespråkar Martin (2009) nio olika riktlinjer som ska hjälpa en programmerare att skriva Clean code. Nedan följer en kort beskrivning av respektive riktlinje för att tydliggöra dess innebörd.

2.2.1 Funktioner

En funktion definieras som en del av ett datorprogram vilken anropas för att utföra en viss uppgift. En funktion skapas ofta i syfte att ha möjlighet att anropas av datorprogrammet vid flera olika tillfällen. (Funktion (programmering), 2017, 11 februari)

Funktioner har ofta en tendens att i slutändan bestå av för många kodrader. Martin (2009) säger att den första regeln man som programmerare behöver tänka på när man skapar en funktion är att den ska vara kort. En kort funktion är lättare att överskåda och förstå för den som läser koden. Han förespråkar även

att en funktion endast ska utföra en uppgift. Martin (2009) menar att ju fler uppgifter en funktion utför, desto lättare är det för fel att ta sig in.

2.2.2 Betydelsefulla namn

Att ha betydelsefulla namn på variabler, funktioner och klasser skulle kunna ses som en grundsten i programmering. Ett betydelsefullt namn gör verkligen skillnad och besvarar många onödiga frågor. (Martin, 2009)

Ett betydelsefullt namn berättar enligt Martin (2009) varför något existerar, vad det ska utföra samt hur det ska användas. Ett namn är inte tillräckligt bra om det behöver förklaras med hjälp av en kommentar. (Martin, 2009)

Nedan följer ett jämförande exempel på hur namngivning av en variabel kan ske. Det första är ett exempel på en mindre bra namngivning medans det efter har ett mer betydelsefullt namn som beskriver variabeln:

```
String fn; // First name of person
```

```
String firstName;
```

2.2.3 Objekt- och datastrukturer

Denna riktlinje behandlar hur en klass bör vara uppbyggd, att den ska gömma sin implementation och gärna vara abstrakt. Martin (2009) förespråkar att det ska finnas interface som döljer klassers implementation och att interface hjälper till att komma åt metoder som finns i de olika klasserna.

2.2.4 Felhantering

Felhantering handlar om vad som ska ske om programkoden av någon anledning inte fungerar som den ska. Så länge felhanteringen inte blir till en störande del i programkoden så är den väldigt viktig att ha med. (Martin, 2009)

2.2.5 Enhetstest

När man som programmerare skriver sin kod så kan det vara viktigt att testa så att den även fungerar som det är tänkt. Något som Martin (2009) skriver om i sin bok är testdriven utveckling (TDD) vilket innebär att ett test skrivs först och efter det programkod. Genom att använda sig av test i sin utveckling så minskar risken för defekter i koden som annars kanske inte skulle upptäckas.

2.2.6 Klasser

Det är viktigt att titta på hur man som programmerare organiserar sina klasser för att även på en högre nivå jobba mot en ren kod. En klass har oftast följande ordning uppifrån och ner (Martin, 2009):

- Publika statiska konstanter
- Privata variabler
- Privata instansvariabler
- Publika funktioner
- Privata funktioner (Martin, 2009)

En viktig sak att tänka på som Martin (2009) nämner är att hålla en klass liten. Han menar att måttet för hur stor en klass bör vara räknas i antal ansvarsområden. Med ansvarsområden menas hur många funktioner en klass innehåller. Ju fler ansvarsområden en klass har, desto svårare är det att förstå vad klassens uppgift egentligen är. (Martin, 2009)

2.2.7 Gränser

Denna riktlinje riktar sig till de tillfällen då det finns en tredje part inblandad. Ett sådant tillfälle kan enligt Martin (2009) vara när man som programmerare använder ramverk från en utomstående part. De som utvecklar ramverk vill att dem ska vara användbara för så många som möjligt vilket kan skapa problem. Dessa ramverk är inte alltid utvecklade på ett sådant vis som förespråkar Clean code vilket såklart påverkar hur programmeraren skriver sin kod. (Martin, 2009)

2.2.8 Formatering

Denna riktlinje riktar sig till hur en programmerare bör formatera sin kod för att den ska bli så läsbar som möjligt. Med formatering menas exempelvis mellanrum mellan rader samt indentering av ny rad.

Enligt Martin (2009) kan formatering av kod ske både vertikalt och horisontellt. Vertikal formatering handlar om att mellan metoder i en klass separera dessa med hjälp av en radbrytning och att kodrader som har något gemensamt bör ligga nära varandra. Vertikal formatering innebär även att det finns en logisk ordning i koden så att man kan förstå ett flöde. (Martin, 2009)

Horisontell formatering innebär istället hur lång en kodrad är eller indentering av rad. En kodrad bör inte vara längre än den behöver. Kodrader kan även ha olika indentering beroende på om kodraden är en start på en klass, funktion eller en variabel som deklareras inuti en funktion. (Martin, 2009)

2.2.9 Kommentarer

När kod inte behöver kommenteras, då är det tillräckligt bra kod. Martin (2009) beskriver kommentarer som ett litet ”misslyckande” som han vill kalla det. Han menar att när en programmerare skriver kommentarer så är grunden till detta att de inte riktigt vet hur de ska uttrycka sig i kod. Kommentarer har också en tendens till att bara lämnas som de skrevs från första början vilket gör att de tillslut blir missvisande. (Martin, 2009)

Betydelsefull och beskrivande namngivning är något som hänger ihop med hur mycket kommentarer som behövs. Behöver man som programmerare kommentera sin kod så är det enligt Martin (2009) dags att titta på koden och kanske ändra på den istället.

2.3 Gemensamt tankesätt

Ett tankesätt kan ses som en uppsättning av metoder eller antaganden som kan användas av en enskild individ eller grupp av människor. (Mindset, 2017, 18 april) Tankesättet hjälper en enskild individ eller grupp att exempelvis etablera ett visst förhållningssätt till något. I denna studie ses förhållningssättet som ett sätt att utföra en slags handling på och handlingen i detta sammanhang innebär att skriva kod. Tankesättet kan även hjälpa till att fortsätta motivera den enskilda individen eller gruppen att fortsätta med tidigare valda metoder. (Mindset, 2017, 18 april)

En tydlig definition av vad ett gemensamt tankesätt kan tänkas bestå utav har inte hittats, utan en modell över vad ett gemensamt tankesätt kan tänkas bestå av utformats enligt nedan. (Se figur 4)

Figur 4. Egen definition av vad ett gemensamt tankesätt består av för delar. (Källa: egen)

Ett gemensamt tankesätt kan tänkas bestå av de tre delarna engagemang, utgångspunkt och överenskommelse. Motiveringen till varför dessa tre delar valts att ingå i definitionen följer nedan.

Den första delen är en utgångspunkt. Med utgångspunkt menas i detta sammanhang uppkomsten till varför detta gemensamma tankesätt valt att etableras. Möjligen om det finns ett tidigare problem eller om det utifrån en viss situiation uppstått något som gör att man väljer att anamma ett nytt tankesätt. I detta sammanhang så är utgångspunkten för det gemensamma tankesättet den tidigare problematiska förvaltningen utav nuvarande TRAP där förvaltningsbarheten av systemen varit mindre bra.

För att ett gemensamt tankesätt ska kunna etableras behövs även engagemang. Engagemang definieras som att de deltagande visar ett intresse för vad som ska utföras och för dess framgång. (Engagemang, 2015, 11 juli) Engagemanget i detta fall innebär ett intresse för tillsammans nå målet att skriva en lättläst och förvaltningsbar kodbas. Engagemanget för det gemensamma tankesättet visas även genom att i teamet använda olika typer av tillvägagångssätt som par- och mobprogrammering men även kodgranskning för att uppnå målet. I tillvägagångssätten som används så förstärks även faktorer som är viktiga för att lyckas med sitt arbete i ett team (se mer om faktorerna, kap 2.6).

Den sista delen är en överenskommelse. Med en överenskommelse menas att de individer eller den grupp som ska etablera ett nytt tankesätt alla är med på det, annars kommer tankesättet inte i slutändan kunna bli ett gemensamt tankesätt. Även i överenskommelsen som ingår i det gemensamma tankesättet kommer tillvägagångssättet in i bilden. Är inte alla i teamet med på det gemensamma tankesättet så kan det möjligtvis vara så att tillämpningen av tillvägagångssättet inte sker som det är tänkt.

2.4 Team

Ett team, arbetsgrupp översatt till svenska, definieras av NE (2017) som en avgränsad grupp av personer som tillsammans arbetar med en gemensam arbetsuppgift. Systemutveckling och systemförvaltning görs i dagsläget oftast i team och kallas då för systemutvecklings- eller systemförvaltningsteam. (Dingsøyr & Dybå, 2012)

2.5 Lättläst och förvaltingsbar kodbas - Icke-funktionella krav

Vid arbete med systemutveckling och även systemförvaltning så brukar det oftast finns en del samlade krav för systemet som ska utvecklas eller förvaltas. Ett systems krav kan delas upp i antingen funktionella eller icke-funktionella krav. Funktionella krav syftar till att beskriva funktioner som ett system bör kunna utföra. Icke-funktionella krav syftar däremot till att beskriva egenskaper ett system bör ha och kan även ses som kvalitetsattribut hos ett system. Dessa kvalitetsattribut kan delas upp i två kategorier: (Non-functional reqiurement, 2017, 26 mars)

- **Exekveringskvaliteér** – Dessa kvalitetsattribut representerar attribut som kan mätas och observeras när systemet körs. Kvalitetsattribut av denna kategori kan vara säkerhet eller användarbarhet. (Non-functional reqiurement, 2017, 26 mars)

- **Evolutionskvaliteér** – Kvalitetsattribut av denna kategori är attribut som är inbäddade i ett systems statiska struktur. Attribut av denna kategori kan vara svåra att mäta och observera jämfört med exekveringskvaliteér. Typiska attribut för denna kategori är förvaltningsbarhet, läsbarhet samt skalbarhet. (Non-functional reqiurement, 2017, 26 mars)

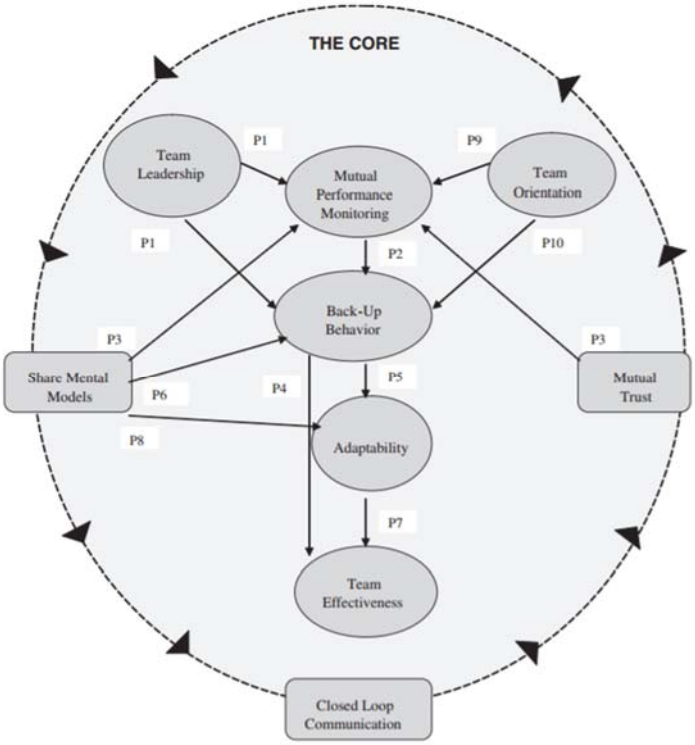
Det finns olika definitioner på vad förvaltningsbarhet är. Förvaltningsbarhet kan bland annat användas som ett kvalitativt mått (Förvaltningsbarhetsindex) för att beskriva hur god förvaltningsbarheten är. (Heitlager, Kuipers & Visser, 2007) Förvaltningsbarhet har dock i detta sammanhang valt att definieras som Rosene et al. (1981) beskriver förvaltningsbarhet vilket är:

”There is a high probability of determining the cause of a problem in a timely manner the first time it occurs, and there is a high probability of being able to modify the program without causing an error in some other part of the program.” - Rosene et al. (1981) (s. 1)

2.6 Big Five in teamwork

Nedan följer en beskrivning av ramverket ”Big Five in teamwork” vilken sammanställts utifrån den vetenskapliga studien *”Is there a ’Big Five’ in teamwork?”* utförd av Salas, Sims & Burke (2005).

Big Five in teamwork är ett ramverk som tagits fram baserat på tidigare studier och analyser. Över 130 modeller har tagits fram under processen som tillslut genererade ramverket. Det innehåller fem dimensioner som tillsammans med tre faktorer anses viktiga för att lyckas med arbete i team. (Se figur 5)



Figur 5. En beskrivning av ramverket "Big five in teamwork" och relationerna mellan de fem dimensioner och tre faktorer som ingår. (Salas, Sims, & Burke, 2005)

Ramverkets fem dimensioner beskrivs enligt följande:

Ledarskap innebär förmågan att leda och samordna arbete och aktiviteter mellan de övriga medlemmar som ingår i teamet, bedöma prestationer, tilldela arbetsuppgifter, utveckla teamets kunskaper och färdigheter samt även motivera och skapa en positiv atmosfär i teamet.

Ömsesidighet kring det arbete som utförs handlar om förmågan att tillämpa lämpliga strategier för att i teamet ha möjlighet att ”övervaka” de arbete som övriga medlemmar i teamet utför. Med det menas att det, inom teamet och mellan medlemmarna som ingår, ska finnas en möjlighet att ge feedback och identifiera misstag som görs utan att se det som något dåligt.

Backup innebär att det finns en kunskap kring de andra medlemmarnas arbete och de ansvar som de har. När kunskapen finns inom teamet bidrar det till en möjlighet att kunna flytta arbetsbelastning mellan olika medlemmar. Viktig faktor att ta hänsyn till vid högbelastade perioder och tidspress för att även under dessa situationer skapa en bra balans i inom teamets arbetsuppgifter.

Anpassningsförmåga handlar om att anpassa strategier baserat på ändrade förhållanden både inom och utanför teamet.

Inriktning innebär benägenheten att beakta allas beteenden under interaktioner i gruppen och att utifrån detta hitta alternativa lösningar och inriktningar. Inriktning innebär även att det finns en tro på att teamets mål är viktigare än den individuella personens mål.

Utöver dessa fem dimensioner har även tre faktorer valts ut som viktiga för ett effektivt arbete i team. Dessa faktorer behövs för att de fem dimensionerna ska fungera på ett bra sätt och de är gemensam mental modell, ömsesidigt förtroende och tvåvägskommunikation och beskrivs kort nedan.

Gemensam mental modell är en modell som bidrar med kunskap i hur teamet ska arbeta och hur medlemmarna ska interagera med varandra. Att kunna identifiera förändringar i ett team eller en uppgift och justera strategier utifrån det.

Ömsesidigt förtroende handlar om att medlemmar i teamet ska dela uppfattningen av att varje medlem tar ansvar över sin roll i teamet och det arbete som ska utföras.

Tvåvägskommunikation innebär att det finns en tvåvägskommunikation vilket inneär att man ska våga fråga om det är något som inte var tydligt nog och att man följer upp för att se att informationen verkligen gått fram.

2.7 Parprogrammering

Parprogrammering är ett tillvägagångssätt som används vid systemutveckling och förvaltning. Sättet innebär att två programmerare skriver kod tillsammans vid en gemensam dator. Den ena ses som förare och skriver koden medans den andra sitter vid sidan om. Detta innebär inte att den ena parten är sysslolös utan att det sker en dialog mellan två programmerare som tillsammans med hjälp av tips och kunskap från varandra försöker programmera på ett bättre sätt. De två som parprogrammerar byter sedan av varandra så att båda får agera förare. (Beck, 2000)

Att använda parprogrammering är ett bra sätt för programmerare att dela kunskap med varandra. Enligt Beck (2000) så kan det hjälpa till att minska kunskapsgap mellan programmerare efter en tids användning.

En annan positiv aspekt med parprogrammering är enligt Beck (2000) att det uppstår färre tillfällen då en programmerare struntar i eller glömmer viktiga saker som man kommit överens om att göra. Programmerare som sitter själva och skriver sin kod har en större tendens till att vid stress eller andra påverkande faktorer strunta i att exempelvis skriva test eller refaktorisera kod. Medans de som parprogrammerar inte gör det. (Beck, 2000)

2.8 Mobprogrammering

Mobprogrammering är ett tillvägagångssätt som kan användas vid både systemutveckling och förvaltning. Detta sätt innebär att hela teamet tillsammans arbetar med samma kod vid samma tidpunkt och även tillsammans vid en gemensam dator. Mobprogrammering kan liknas vid parprogrammering då det endast finns en såkallad förare. Föraren skriver kod medans de övriga i teamet samarbetar och diskuterar för att tillsammans generera bra kod. Vid mobprogrammering används en timer för att i intervall byta föraren som skriver kod. (Agile Alliance, 2015)

2.9 Kodgranskning

Kodgranskning, på engelska code review, är ett sätt som används för att hitta buggar och misstag som finns i kod som skrivits. Kodgranskning kan ske manuellt då en programmerare läser koden och letar efter buggar och misstag. Granskningen kan även göras med hjälp av olika verktyg. Det är ganska vanligt att kodgranskning görs i samband med parprogrammering eftersom två programmerare arbetar ihop och samtidigt granskar varandras kod. Det kan även användas när en programmerare skrivit kod enskilt och sedan vill ha den granskad av en annan programmerare. (Code Review, 2017, 28 april)

Cohen et. al (2006) menar att kodgranskning borde vara en del av systemutvecklingsprocessen. En jämförande studie visar att kodgranskning i systemutvecklingsprocessen kan hjälpa till att minska hälften av de kostnader som uppstår för att reparera buggar i koden. Enligt Cohen et. al (2006) finns även fördelar så som att kodgranskning hjälper till att öka kodkvaliteten och även kommunikationen kring den kodbas som utvecklas eller förvaltas. Arbetssättet kan även vara till hjälp för att skapa förståelse och utbilda juniora programmerare. (Cohen et. al, 2006)

2.10 Tidigare forskning

Ingen tidigare forskning kring team och Clean code i kombination med varandra har hittats. Däremot har en tidigare studie om forskning kring teamwork och de faktorer som ingår i ramverket ”Big five in teamwork” hittats som kan vara av intresse för denna studie.

Studien är utförd av Andreassen (2015) och fokuserar på att undersöka agil systemutveckling i storskaliga team. I denna studie användes det ramverk som presenterades i det tidigare avsnittet Big Five in teamwork (3.6). Resultatet från denna studie tyder på att ömsesidigt förtroende och gemensam mental modell är två faktorer som är viktiga för att storskaliga team ska lyckas med agil systemutveckling. (Andreassen, 2015)

En annan studie vars resultat kan vara av intresse för denna studie är en som genomförts av Mathieu et. al (2000). Studiens syftade till att undersöka om gemensamma mentala modeller för hur team ska arbeta gav en positiv effekt på dess prestationer. Resultatet från denna studie visar på att en gemensam mental modell för hur ett arbete ska utföras har en positiv påverkan på ett teams prestationer. Mathieu et. al (2000) En av de betydande faktorerna som ingår i ramverket ”Big five in teamwork” är just gemensam mental modell. (Salas, Sims & Burke, 2005)

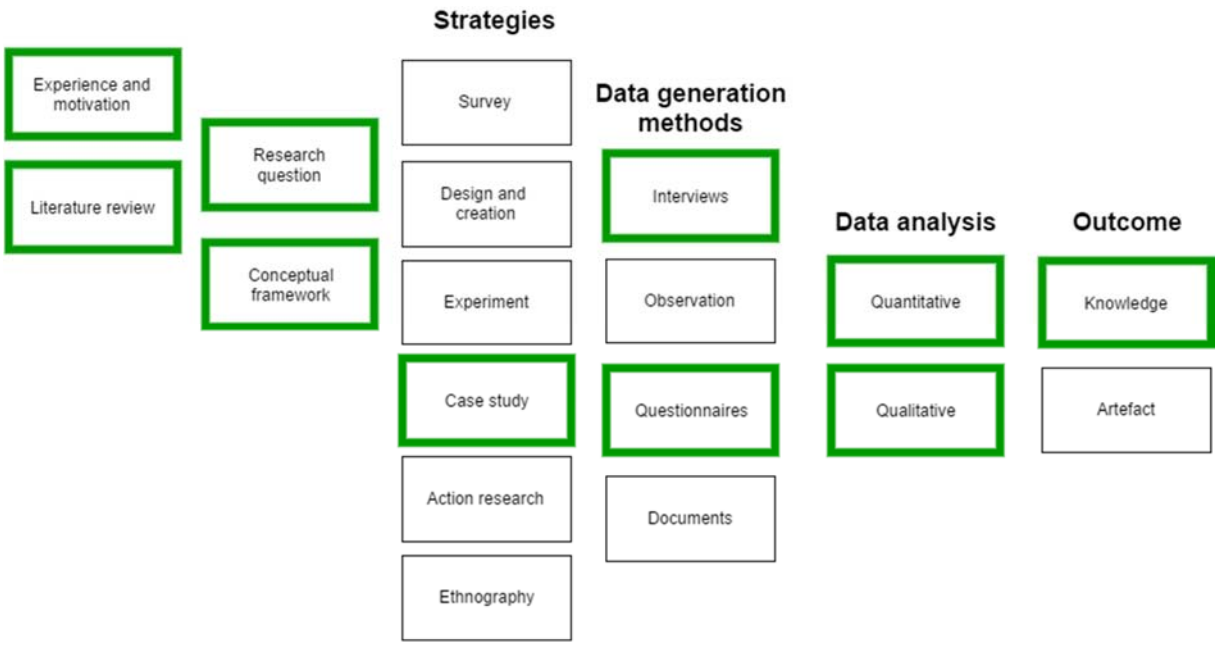
Utifrån resultatet från tidigare forskning så kändes det intressant att använda sig av ramverkvet ”Big five in teamwork” i denna studie men att istället rikta det mer specifikt mot team och det gemensamma tankesättet som grundas i Clean codes riktlinjer.

3. Metod

I detta kapitel beskrivs studiens genomförande där bland annat val av forskningsstrategi (3.3), datainsamlingsmetoder (3.5) samt dataanalys (3.8) presenteras och motiveras.

3.1 Forskningsprocessen

Studien har genomförts enligt Oates (2006) forskningsprocess som illustreras i figuren nedan (figur 6) där val av strategi, datainsamlingsmetod och dataanalys har anpassats för att på bästa sätt uppfylla studiens syfte. De val som gjorts för denna studie är grönmarkerade och kommer beskrivas mer ingående under kommande avsnitt i detta kapitel. En studie genererar alltid ett resultat och enligt Oates (2006) så består resultatet av antingen en artefakt eller kunskap. Det som i slutändan har genererats från denna studie är kunskap.



Figur 6. Modell över forskningsprocessen. (Oates, 2006, s.33)

3.2 Abduktion

Vilken ansats ett arbete har beror på vilken ändpunkt arbetet startat i. Björklund och Paulsson (2012) beskriver att under tiden en uppsats skrivs, vandrar man mellan två olika abstraktionsnivåer. De två abstraktionsnivåerna som även utgör utgör ändpunkterna är teorin (hög abstraktionsnivå) och empirin (låg abstraktionsnivå).

- En induktiv ansats innebär att studien startar från verkligheten, empirin, och utifrån den kan sedan mönster sammanfattas och sättas ihop till modeller och teorier som är användbara för studien. (Björklund & Paulsson, 2012)
- En deduktiv ansats innebär att studien utgår från en teori och att förutsägelser kring empirin görs baserat på teorin. (Björklund & Paulsson, 2012)

- Abduktion innebär att en kombination av dessa sker och man vandrar fram och tillbaka mellan dessa abstraktionsnivåer. (Björklund & Paulsson, 2012)

I denna studie så har en abduktion skett. Inför studien så genomfördes litteraturstudier för att skapa en teoretisk referemsram för arbetet. Den teoretiska referensramen var till hjälp vid utformning av vissa intervjufrågor och även enkäten då den byggde på ramverket Big Five in teamwork (2.6). När datainsamling skett och empiri sammanställts så upptäcktes ytterligare begrepp som ansågs ha en plats i den teoretiska referensramen inför kommande analys. I och med detta så har studien baserats på både teori och empiri, vilket innebär att både deduktion och induktion skett för att slutligen generera ett resultat. Eftersom en vandring mellan dessa abstraktionsnivåer gjorts under studien så har en abduktion skett.

3.3 Forskningsstrategi

En forskningsstrategi är det som representerar den övergripande strategin för hur en studie ska genomföras. Forskningsstrategin ska understöjda forskaren att besvara de forskningsfrågor som studien grundas på. (Oates, 2006)

Den forskningsstrategi som valts att användas för den denna studie är en fallstudie. En fallstudie används när man vill fokusera på ett specifikt fall vilket kan vara en utvald avdelning hos en verksamhet, en verksamhetsprocess eller som i denna studie ett specifikt team hos ett IT-företag. Resultatet av en fallstudie ska i viss mån gå att generaliseras. (Oates, 2006) Resultatet från denna studie har bidragit med kunskap i form av hur ett team arbetar med att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer men även betydande faktorer att beakta i en sådan situation. Den genererade kunskapen kan vara av intresse för andra team som står inför nyutveckling eller för team som arbetar med förvaltningsuppdrag. Resultatet kan även vara av intresse rent generellt då den bidrar med nyttig kunskap kring viktiga faktorer att beakta.

Anledningen till varför denna forskningsstrategi valt att användas är för att skapa en detaljrik och beskrivande bild över det specifika fall som ska studeras, i detta fall teamet på Nethouse i Borlänge. Enligt Oates (2006) genererar en fallstudie en djupare förståelse för det fall som valts att studeras då denna forskningsstrategi förespråkar olika datainsamlingsmetoder. Både kvalitativ och kvantitativ datainsamlingsmetod har använts i undersökningen i form av intervjuer och enkäter.

En annan forskningsstrategi som varit möjlig att använda för studien hade varit design and creation-strategin. Dock genererar en design and creation-baserad studie i grund och botten en artefakt. (Oates, 2006) Denna studie syftade till att ge kunskap kring ämnet clean coding i team och även betydande faktorer i ett sådant sammanhang och inte till att sammanställa exempelvis riktlinjer eller ett ramverk för hur ett team bör gå tillväga vid etableringen.

Deskriptiv och normativ fallstudie

Enligt Björklund och Paulsson (2012) så kan en studie vara av fyra olika typer beroende på vilken kunskapsmängd som finns inom området.

- **Explorativ** – Denna typ av studie utförs när det finns ingen eller endast lite kunskap inom området. En studie av denna typ ser till att skapa en grundförståelse för området som studeras.
- **Deskriptiv** – Även kallat för beskrivande, är en typ av studie som används då det redan finns grundläggande kunskap inom området och målet endast syftar till att beskriva och inte förklara.
- **Explanativ** – När studien syftar till att uppnå djupare kunskap och förståelse inom ett område och målet är både att beskriva och förklara.

- **Normativ** – När det redan finns förståelse och kunskap inom ett kunskapsområde används en normativ studie. En normativ studie syftar till att ge vägledning och föreslå åtgärder. (Björklund & Paulsson, 2012)

Denna studie kan till viss del ses som en deskriptiv studie i och med att det redan finns kunskap inom områdena Clean code, tankesätt och team. Den första frågan i frågeställningen handlar om att beskriva hur teamet arbetar med etableringen av det gemensamma tankesättet idag och detta tyder också på att det är en deskriptiv studie. Studien kan även ses som en normativ studie då den andra frågan i frågeställningen syftar till att ta reda på vilka faktorer som kan anses vara viktiga när ett nytt gemensamt tankesätt ska etableras vilket kan anses vara en viss typ av vägledning.

3.4 Litteraturstudier

Litteraturstudier utförs enligt Oates (2006) vanligtvis i två steg. Första steget innebär att med hjälp av litteraturstudier inom ämnet hitta ett relevant område att undersöka och även här finna forskningsideér. Andra steget är en mer ingående litteraturstudie inom ämnet. (Oates, 2006) En styrka med litteraturstudier är enligt Björklund och Paulsson (2012) att man som forskare under en kort period kan få hjälpa att samla in mycket information kring ämnet.

Första steget i litteraturstudien gjordes i syfte att ta reda på om det fanns tidigare relevant forskning inom området som behandlar Clean code och gemensamma tankesätt i team i kombination. Det senare steget genomfördes i syfte att skapa en djupare förståelse för både Clean code och gemensamma tankesätt samt även för att skapa en användbar referensram för studien.

För att genomföra litteraturstudien har databaser som Google Scholar, IEEE Xplore Digital Library samt högskolans egna databas Summon använts. Databasen DiVA (Digitala Vetenskapliga Arkivet) har även använts och är en databas vilken bland annat omfattar vetenskapliga arbeten från olika högskolor och universitet inom Sverige. I Summon har filtreringsfunktionen vetenskapligt granskade artiklar använts för att hitta trovärdiga källor. Vid sökning i Google Scholar har en medveten källkritisk granskning utförts genom att bland annat titta på tidsskrift som artikeln publicerats i och vilka/vilken författare som skrivit artikeln och sedan utifrån egna erfarenheter bedöma om källan verkar trovärdig. Antal citeringar som gjorts på en artikel har även använts för att avgöra om källan verkar trovärdig.

En litteratursökning bör enligt Oates (2006) vara genomgående strukturerad och för att skapa denna struktur användes en konceptbaserad litteratursökning. En sådan sökning innebär att en mening som representerar det man söker utformas för att sedan delas upp i en såkallad koncepttabell. Utifrån egen erfarenhet av tidigare litteratursökning så valdes en mening på engelska eftersom det oftast genererar fler träffar:

How can a team manage to implement a common mindset that
will help them to create a code base that is readable and easy to maintain?

Denna mening delades upp i koncept och fördes sedan in i en koncepttabell. (se tabell 1). Koncepten har sedan använts i kombination för att hitta relevant litteratur.

Team	**Implementation**	**Mindset**	**Readable**	**Maintenance**
Group Employees Collaborate	Implement	Way of thinking Holistic Paradigm	Understandable Legible Comprehensible	Management

Tabell 1. Koncepttabell enligt Oates (2006) förslag på konceptbaserad litteratursökning.

Utöver dessa koncept så har även andra synonymer använts för att bredda sökningen än mer. Dessa synonymer är nyckelord som inte passat in i koncepttabellen eller som hittats när sökningar grundade på koncepttabellen utförts.

- Code principles
- Clean code
- Development team Clean code
- Difficulties implementation Clean code
- Maintenance improvement Clean code

Både backward och forward search har använts som ytterligare metoder vid litteratursökningen. Enligt Webster et. al (2002) går backward search ut på att i en relevant källa granska referenserna för att se om det i referenserna finns ytterligare källor som kan vara relevanta för studien. Genom att använda backward search har flest källor hittats som varit relevanta för studien. När en forward search görs så utgår man från en relevant källa och väljer att titta på andra källor som refererat till den relevanta källan. (Webster & Watson, 2002) För att göra en forward search har vetenskapliga sökmotorn Google Scholar använts.

Baserat på den litteratursökning som gjorts så har ingen tidigare forskning hittats som handlar om just hur ett team kan gå tillväga för att införa Clean code som ett gemesamt tankesätt. Denna upptäckt kan såklart ses som både positiv och negativ. Det kan vara positivt att det saknas kunskap kring området då den nuvarande studien kan hjälpa till att börja fylla detta kunskapsgap. Det negativa är att det varit svårt att hitta lämpliga modeller och metoder för utförandet av studien eftersom det inte funnits någon tidigare studie att luta sig mot.

3.5 Datainsamlingsmetod

För att generera primärdata till undersökningen så används antingen en eller flera datainsamlingsmetoder. Några av de vanligaste datainsamlingsmetoder som används i forskningssyfte idag är intervjuer, observationer, enkäter samt dokumentstudier. (Oates, 2006) Dessa metoder kan enligt Oates (2006) delas upp i kvalitativa och kvantitativa datainsamlingsmetoder.

- **Kvalitativ datainsamlingsmetod** syftar till att generera data i form av ord, bilder och ljud.
- **Kvantitativ datainsamlingsmetod** syftar till att generera numerisk data. (Oates, 2006)

För denna studie krävdes två datainsamlingsmetoder för att uppfylla studiens syfte. En kvalitativ datainsamlingsmetod i form av intervjuer samt en kvantitativ som utgjordes av en enkät.

3.5.1 Intervjuer

Intervjuer är en kvalitativ datainsamlingsmetod vilken kan utföras i tre olika former, antingen genom strukturerade, semi-strukturerade eller ostrukturerade intervjuer. (Oates, 2006) En strukturerad intervju används då ett fastställt frågeformulär finns och när det finns en viss ordning på frågorna så som de bör ställas. Vill man att det ska finnas plats för att ställa frågor som dyker upp under intervjutillfället bör man använda en semi-strukturerad intervju. Denna form av intervju utgår från endast några fastställda frågor. Den tredje formen av intervju kallas ostrukturerad intervju och innebär att alla frågor uppkommer under intervjutillfället. (Björklund & Paulsson, 2012)

Den form av intervjuer som lämpade sig bäst för denna studie var semi-strukturerade intervjuer. Detta för att det fanns en del frågor som krävde svar men även för att vara öppen för ytterligare frågor om det skulle dyka upp några under intervjutillfället.

Intervjufrågorna (se bilaga 1) utformades för att ha möjlighet att svara på frågorna:

- Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?
- Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

Fem intervjuer genomfördes på plats hos Nethouse. Intervjufrågorna delades upp i tre delar eftersom de fokuserade på olika områden. Första delen användes för samla in generell information kring respondentens åsikt gällande team och programmering men även för att få svar på vilka faktorer respondenten ansåg vara viktiga vid teamwork. Första delen avsåg att hjälpa till att besvara frågan: *”Vilka faktorer i ett team kan anses viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?”*.

Andra delen användes för att få svar på hur respondenten tänker kring det gemensamma tankesättet som grundas i Clean code riktlinjer. Den tredje och sista delen av intervjufrågorna fokuserade på att ta reda på hur teamet arbetar idag för att etablera tankesättet och tillämpa Clean codes riktlinjer. Del två och tre användes för att hjälpa till att besvara frågan: *”Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?”*.

Intervjuerna spelades in för att på bästa sätt ha möjlighet att vid transkribering återspegla det som respondenten sagt under intervjun. (Se bilaga 3)

3.5.2 Enkäter

En enkät användes som den kvantitativa datainsamlingsmetoden för studien. En enkät är enligt Oates (2006) en samling fördefinierade frågor som är skrivna i en viss ordning. Enkäten kan innehålla olika typer av frågor, bland annat frågor där respondenten kan ombes att rangordna faktorer enligt en viss numrering. (Oates, 2006)

Enkäten bestod av endast en fråga som var baserad på ramverket Big Five in teamwork. Ramverket förespråkar fem dimensioner och tre faktorer som anses vara viktiga för att ett team ska fungera bra. Dessa är ledarskap, ömsesidighet kring arbete som utförs, back-up, anpassningsförmåga, inriktning, gemensam mental modell, ömsesidigt förtroende och tvåvägskommunikation. (Salas, Sims och Burke (2005) I denna studie har dimensionerna ur ramverket valt att benämnas för faktorer eftersom de faktorer som i slutändan tagits fram i slutsatsen inte endast kommer från detta ramverk.

Respondenten ombads att rangordna faktorerna från 1-8, där 1 är mycket viktig medans 8 är mindre viktig utifrån perspektivet där ett team ska etablera ett nytt gemensamt tankesätt som grundas i Clean codes riktlinjer. Enkäten (se bilaga 2) skickades ut via mejl till samma respondenter som deltagit i intervjuerna som ett kompletterande intervjutmaterial för att ha möjlighet att jämföra och ge ytterligare underlag till att svara på frågan:

- Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

SurveyMonkey (sv.surveymonkey.com) användes som verktyg för att utforma enkäten och även samla in och sammanställa svaren.

3.5.3 Urval av personer till datainsamling

Urvalet av respondenter till datainsamlingen gjordes tillsammans med en av mina handledare på Nethouse. Vi kom fram till att det vore mest intressant för studien att välja respondenter med olika erfarenhet av att arbeta i teamet samt med någon typ av skillnad i roll. Med erfarenhet menas hur länge respondenterna arbetat i teamet. Med roll menas att alla respondenter har rollen systemutvecklare men att det skiljer sig till viss del då en av respondenterna har en ledande utvecklarroll jämfört med övriga. Antal respondenter som deltog i studien var totalt fem stycken.

3.6 Metodtriangulering

För att höja studiens tillförlitlighet kan flera olika metoder användas för att samla in data. (Oates, 2006) Enligt Björklund och Paulsson (2012) så kallas detta för metodtriangulering och innebär att flera olika metoder används för att undersöka en och samma företeelse. I denna studie har både intervjuer och även en enkät använts för att uppnå studiens syfte samt öka studiens tillförlitlighet. Triangulering har använts för att i den mån det varit möjligt jämföra intervjusvar mot enkätsvar i syfte att analysera betydande faktorer i ett team vid etableringen av ett gemensamt tankesätt som grundas i Clean codes riktlinjer.

3.7 Forskningsetik

Det är viktigt att ta hänsyn till samtliga personer som är involverade i en studie som genomförs. Typiska och direkt involverade personer är respondenter som medverkat i enkätundersökningar och intervjuer i syfte att samla in data som ska stödja undersökningen. (Oates, 2006)

I studien så genomfördes både intervjuer och enkäter. Som genomförare av studien ville jag vara säker på att utförandet av datainsamlingen skedde enligt de riktlinjer som Oates (2006) förespråkar när de kommer till forskningsetik:

- Respondenten har rätt att tacka nej till att genomföra en intervju eller enkätundersökning även fast denne tidigare tackat ja till att medverka. (Oates, 2006)
- Respondenten har rätt att bli korrekt informerad om studiens syfte och hur det insamlade materialet kommer användas. (Oates, 2006)
- Respondenten har rätt till att vara anonym. (Oates, 2006)
- Den som genomför studien har en skyldighet till respondenten att förvara det insamlade materialen konfidentiellt. (Oates, 2006)

Respondenterna som deltagit i studien har informerats om studiens syfte. De har även informerats om att de förblir anonyma och att den enda som har tillgång till det insamlade data är jag som utför studien och att det endast kommer användas för denna studies syfte.

3.8 Dataanalys

Data som genereras från datainsamlingen behöver också analyseras. Det finns enligt Oates (2006) två sätt att analysera data och sättet man analyserar på bestäms av vilken typ av data som samlats in. En kvalitativ dataanalys utförs på kvalitativ data som samlats in vilket kan vara transkriberade intervjuer. En kvantitativ dataanalys utförs på kvantitativ data vilket kan vara resultatet från en enkät i form av siffror. (Oates, 2006)

Två typer av datainsamlingsmetoder har använts där intervjuerna (2.4.1) genererat kvalitativ data i form av transkriberade intervjuer medans den enkät (2.4.2) som använts genererat kvantitativ data i form av siffor.

Enligt Oates (2006) så bör det kvalitativa data som ska analysera först förberedas för att underlätta vid den kvalitativa analysen. Intervjuerna transkriberades och sammanställdes i varsitt word-dokument för att skapa liknande struktur för att underlätta vid analysen. Intervjufrågorna var redan från början uppdelade i tre olika delar vilket också underlättade vid den kvalitativa analysen.

Nästa steg i en kvalitativ dataanalys är enligt Oates (2006) att identifiera teman, samband eller mönster vilket kan göras på ett induktivt eller deduktivt sätt. Detta steg förklaras mer ingående under kommande avsnitten (3.8.1 och 3.8.2) i detta kapitel.

3.8.1 Kvalitativ nulägesanalys

En nulägesanalys utförs i syfte att få en objektiv bild över hur ett företag eller team fungerar just nu. (Nulägesanalys, 2015, 27 maj) En kvalitativ nulägesanalys (5.2) utfördes för att ha möjlighet att besvara den första frågan i frågeställningen:

- Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?

Del två och tre av de transkriberade intervjuerna lästes igenom för att skapa en uppfattning kring vad respondenterna sagt. Nästa steg i analysen innebar att kategorisera upp dessa delar av de transkriberade intervjuerna i följande kategorier:

- **Tillvägagångssätt** - Hur går teamet till väga för att sprida kunskap kring det gemensamma tankesättet som grundas i Clean codes riktlinjer.
- **Tillämpningen av det gemensamma tankesättet** – Vilka riktlinjer i det gemensamma tankesättet som tillämpas idag.
- **Förhållningssättet till det gemensamma tankesättet** – Vilken attityd som finns inom teamet till det gemensamma tankesätt som grundas i Clean codes riktlinjer.

Dessa kategorier har valts ut induktivt utifrån de transkriberade intervjuerna för att de tillsammans skapar en helhet över arbetet idag. Indelningen i kategorierna bidrar till att hjälpa mig besvara frågan som syftar till att ta reda på hur arbetet med etableringen av det gemensamma tankesättet ser ut idag.

I tabell 2 nedan så har jag illustrerat ett exempel på hur jag gjort när jag kategoriserat in de transkriberade intervjuerna i de tre olika kategorierna som ingår i nulägesanalysen.

Tillvägagångssätt	**Tillämpningen av det gemensamma tankesättet**	**Förhållningssättet till det gemensamma tankesättet**
".. vi parprogrammerar och mobprogrammerar och det är delvis med en i teamet då som är väldigt duktig på det."	" Ser jag någon som gör mer än en sak så bryter jag isär den. Det är ju boy scout rule, den kör vi. Och lika försöker vi ge dem väl beskrivna namn. Funktionsnamn och klassnamn. "	" Det ska vara gemensamt ägande på kodbasen så att vem som helst kan ändra vart som helst." "Det är ändå teamet som ska kunna ändra på grejerna."

Tabell 2. Exempel på hur uppdelning i kategorier gjorts i nulägesanalysen.

I tabell 3 nedan så har ett exempel illustrerats på hur empirin jämförts med teorin.

Respondents svar	**Teorin säger**
".. vi parprogrammerar och mobprogrammerar och det är delvis med en i teamet då som är väldigt duktig på det."	Parprogrammering är ett ypperligt sätt att dela kunskap med andra. (Beck, 2000) Parprogrammering kan hjälpa till att minska kunskapsgap mellan programmerare efter en tids användning. (Beck, 2000)

Tabell 3. Exempel på hur emipri jämförts med teori i den kvalitativa nulägesanalysen.

I tabell 4 nedan så har även ett exempel illustrerats över hur tolkning av empirin gjorts i nulägesanalysen.

Respondents svar	**Tolkning**
”Det är ju viktigt att alla gör på samma sätt också” ” ... att vi kommer skriva någorlunda lika och att göra på samma sätt så att inte systemet eller systemen är byggda eller skriva på massa olika sätt.”	Det gemensamma tankesättet som grundas i Clean codes riktlinjer ses som något positivt eftersom det är ett sätt att få medlemmarna i teamet stt skriva sin kod på liknande sätt.

Tabell 4. Exempel på hur empirin har tolkats i nulägesanalysen.

3.8.2 Analys av betydande faktorer i team

Både kvalitativ- och kvantitativ dataanalys har gjorts i syfte att besvara andra frågan:

- Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

Den kvalitativa dataanalysen utfördes på de transkriberade intervjuerna. Intervjuerna lästes igenom och fokus låg på del ett av de transkriberade intervjuerna eftersom den var utformad att behandla vilka faktorer som ansågs vara viktiga vid teamwork. Faktorerna som nämnts i intervjusvaren sammanställdes och rangordnades där 1 ansågs vara den viktigaste faktorn. (se figur 11, kap 5.2)

Enkätsvaren sammanställdes i ett diagram med hjälp av enkätverktyget SurveyMonkey. Enkäten genererade kvantitativ data i form av ordinaldata. Ordinaldata syftar till att beskriva en viss ordning eller ranking på det som avser att undersökas. (Oates, 2006) Enkätverktygen SurveyMonkey hjälpte mig att presentera resultatet av den ordinaldata som genererats genom att omvandla den till poäng. Den kvantitativa analysen genomfördes då faktorerna sammanställdes och rangordnades manuellt efter den poäng de fått enligt diagrammet. (se figur 11, kap 5.2)

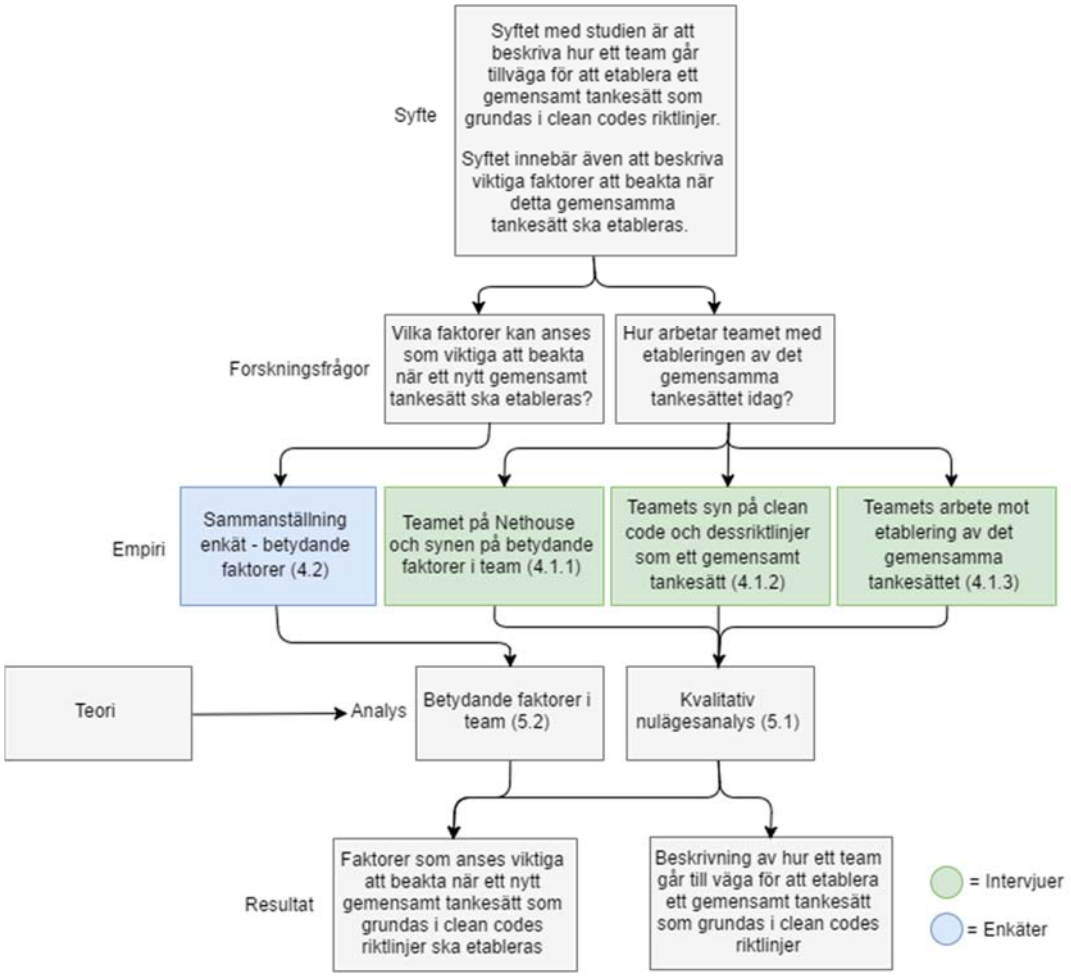
Rangordningen som sammanställts från intervjusvaren jämfördes sedan med enkätsvaren. Jämförelsen gjordes i syfte att se om det fanns någon skillnad i valet av faktorer beroende de olika perspektiven. Den utfördes även för att utifrån detta kunna dra slutsatser om vilka faktorer urifrån ramverket Big Five in teamwork som kan anses vara viktiga vid etablering av ett nytt gemensamt tankesätt i ett team.

3.9 Genomförande

Nedan så har en övergripande bild ritats för att förklara hur studien genomförts. (se figur 7) Syftet som studien byggt på har vart att beskriva hur ett team går till väga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer. Studien har även byggts på syftet som innebar att beskriva vilka faktorer som kan anses som viktiga att beakta när ett nytt gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras.

Forskningfrågorna som använts i studien har varit till hjälp för att bestämma vilka datainsamlingsmetoder som använts. Intervjuer och enkäter som genererat primärdata till studien har resulterat i empiri som använts i analysen. I analysen har sedan empiri analyserats, bekräftats och jämförts mot teorin.

Utifrån analyserna så har sedan en beskrivning av hur ett team går tillväga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer gjorts. Även en slutsats har dragits gällande vilka faktorer som anses vara viktiga att beakta när ett nytt gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras.



Figur 7. Övergripande bild över genomförandet av studien.

4. Empiri

I detta kapitel presenteras genererat data från datainsamlingsmetoderna som använts. Intervjuerna presenteras i de tre delarna: teamet på Nethouse och synen på betydande faktorer i team (4.1.1), teamets syn på Clean code och dess riktlinjer som ett gemensamt tankesätt (4.1.2) och teamets arbete mot etablering av det gemensamma tankesättet (4.1.3). Enkäten presenteras med hjälp av tabell och diagram (4.2)

4.1 Sammanfattning av intervjuer

Det data som samlats in med hjälp av intervjuer har sammanfattats och presenteras först i en kort beskrivande text och därefter följer en sammanfattning av respektive intervjufråga i tabellform. Intervjuerna presenteras i tre delar vilka speglar de tre delar som intervjufrågorna delats upp i.

4.1.1 Teamet på Nethouse och synen på betydande faktorer i team

TRAP-teamet på Nethouse är ett tämligen nytt team vilket startades för ca 2 år sedan. Teamet består av relativt nyutexaminerade studenter men även en medlem med lite mer erfarenhet sedan tidigare. De har alla en positiv uppfattning av att arbeta i team och tycker att det finns fler fördelar än nackdelar med att arbeta i team. Exempel på fördelar som pekats på är att det alltid finns hjälp nära till hands och att man tillsammans arbetar med en gemensam kodbas. Något som ses som viktigt hos respondenterna är att det finns ett gemensamt synsätt eftersom de arbetar med samma kodbad och mot samma mål.

I tabellen nedan (se tabell 5) presenteras en sammanställning av svaren på respektive intervjufråga som ingick i del ett. (Se bilaga 1)

Fråga 1. Hur länge har du jobbat med programmering?
4 av 5 respondenter har arbetat med programmering i ca 1-2 år då dessa är relativt nyutexaminerade studenter. En utav teammedlemmarna har däremot arbetat betydligt längre, sedan -92/-93. Den senaste har även rollen som leadutvecklare i teamet.
Fråga 2. Hur länge har du jobbat i teamet på Nethouse?
Fyra av respondenterna har arbetat i teamet i ca 1,5 – 2 år, sedan TRAP-teamet startades. En respondent som nyligen anslutit sig till teamet.
Fråga 3. Vilka är dina uppfattningar av att jobba i team? Positiva och negativa.
Övervägande svar är att det alla har positiva uppfattningar av att arbeta i team. Att man tillsammans kan lösa problem, inte behöver hålla koll på allting själv. Det blir inte samma press på en person utan hela teamet får dela på den tillsammans. Det finns alltid någon nära till hands om man behöver hjälp med något. Negativt kan vara om det uppstår konflikter eller när man behöver koncentrera sig då det ofta kan bli en hel del prat. Men detta brukar lösas genom att använda hörlurar för att koppla bort sådant. Om nya arbetssätt eller liknande ska bestämmas så kan det ta lite längre tid när man är ett team, eftersom alla ska vara överens innan man bestämmer sig för att köra på något.

Fråga 4. Det finns 8 uttalade faktorer som anses vara viktiga för att ett team ska fungera bra och dessa är:

- **Ledarskap** – att det finns någon person som ses som mer styrande i teamet.
- **Ömsesidighet kring det arbete som utförs** - med detta menas att det är högt i tak och att det känns okej att lägga sig i andra teammedlemmars kod och även tvärt om.
- **Backup** - att det finns en plan för hur teamet går till väga för att klara tungt belastade perioder.
- **Anpassningsförmåga** - hur lätt teamet och de individer som ingår har för att anpassa sig till nya tankesätt och arbetssätt.
- **Inriktning** - en inriktning med tydligt mål finns definierat inom teamet.
- **Gemensam mental modell** – kan ses som en modell som bidrar med kunskap i hur teamet ska arbeta och hur medlemmarna ska interagera med varandra.
- **Ömsesidigt förtroende** - handlar om att medlemmar i teamet ska dela uppfattningen av att varje medlem tar ansvar över sin roll i teamet och det arbete som ska utföras.
- **Tvåvägskommunikation** - innebär att det finns en kommunikation mellan teammedlemmar och att man följer upp för att se att informationen verkligen gått fram.

Vilka tre av dessa åtta faktorer anser du är viktigast för ett team? Varför?

Ledarskap **IIII**
Ömsesidighet kring det arbete som utförs **III**
Backup
Anpassningsförmåga **III**
Inriktning **I**
Gemensam mental modell
Ömsesidigt förtroende **IIIII**
Tvåvägskommunikation **II**

Ömsesidigt förtroende är den faktor som fem av respondenterna valt att ange. Fyra av fem respondenter valde faktorn ledarskap. Ömsesidighet kring de arbete som utförs samt anpassningsförmåga har valts av lika många respondenter. Inriktning och tvåvägskommunikation har även fått varsin röst. Back-up är en faktor som ingen respondent valt att ta upp som en av de tre viktigaste.

Fråga 5. Vad anser du är viktigt när det kommer till team och programmering i kombination med varandra?

Viktigt är att tillsammans ha ett gemensamt synsätt eftersom alla i teamet jobbar tillsammans och med samma kodbas. Det är även viktigt att sprida kunskap inom teamet på ett bra sätt. Att våga fråga när man är osäker på någonting eller när man inte förstår kod, att man i teamet stället upp för varandra när det behövs hjälp och att man kan bolla ideér med varandra.

Fråga 6. Tycker du att det finns någon problematik att som programmerare koda i ett team? Varför?

Att de som ingår i teamet tänker olika när det kommer till programmering och att man då måste hitta ett sätt att lösa detta på för att kunna arbeta som ett team. Om det finns olika kunskapsnivåer i teamet kan det kanske vara ett litet problem, detta kan dock se till att lösas genom kommunikation vilket är viktigt i ett sånt här sammanhang.

Tabell 5. Sammanfattning av intervjuer, del 1.

4.1.2 Teamets syn på Clean code och dess riktlinjer som ett gemensamt tankesätt

Begreppet Clean code som används inom teamet är det som definieras av Robert C. Martin som även beskrivs i teoriavsnittet. Det finns en förståelse för begreppet och vilka riktlinjer som ingår. Respondenterna påpekar dock att detta kan läsas på mer om för att skapa en ännu bättre förståelse. Respondenterna har innan tänkt lite på hur de skriver sin kod men medger att de efter att de fått kunskap kring Clean codes riktlinjer tänker mer på hur de skriver sin kod.

Clean code är enligt respondenterna viktigt då de beskriver att man i framtiden inte vet vem eller vilka som kommer arbeta med koden som man lämnar ifrån sig. En annan viktig sak som respondenterna påpekar är att man inte vill bli beroende av någon specifik person i teamet för att man skriver kod på olika vis.

De tillfrågade teammedlemmarna har inte tidigare arbetat enligt ett uttalat sätt att skriva sin kod på. En aspekt som tas upp är att dessa riktlinjer inte ska ses som regler då de anser att det istället kan krångla till det hela. Även att man bör komma överens om vilka riktlinjer som man ska följa.

För att sprida kunskapen kring det gemensamma tankesätt som grundas i Clean codes riktlinjer beskrivs parprogrammering och mobprogrammering som sätt för att göra detta. Att man även ger varandra tips som man hittat eller liknande är också ett sätt som används för att sprida kunskapen.

I tabellen nedan (se tabell 6) presenteras en sammanställning av svaren på respektive intervjufråga som ingick i del två. (Se bilaga 1)

Fråga 1. Har du tidigare hört talas om begreppet Clean Code och vet vad det omfattar?
Alla respondenter har hört talas om begreppet Clean code tidigare. De flesta vet vad begreppet omfattar och vilka riktlinjer Uncle Bob förespråkar när han pratar om Clean code. Några har och håller på att läsa böcker om clean coding för att lära sig mer om allt det omfattar.
Fråga 2. Har du tidigare tänkt på hur du skriver din kod?
Respondenterna har tidigare försökt tänka åtminstone lite grann på hur de skrivit sin kod. Dock har de börjat tänka mer på detta efter att de börjat läsa och lära sig mer om Clean code och dess riktlinjer. Tidigare så fanns andra riktlinjer för hur man borde skrivit sin kod, men att detta också är trender som byts ut.
Fråga 3 & 4. Tycker du att det är viktigt att skriva Clean Code? Varför tycker du att det är viktigt att skriva Clean Code?
Anledningen till varför det är viktigt att skriva Clean code är enligt respondenterna att man inte i framtiden vet vem som kommer titta på koden man skriver. Att man på ett så bra sätt som möjligt vill göra koden begriplig. Sen även att man tillsammans arbetar i ett team, gemensamt ägande av kodbasen. Att vem som helst ska kunna hoppa in i koden och förstå och att inte behöva bli beroende av någon specifik person i teamet. Viktigt med tydlig och välskriven kod inför framtida förvaltning av kodbasen.
Fråga 5. Har du någonsin tidigare jobbat enligt ett utalat sätt att skriva din kod på? Eller har det mer varit att var och en sköter sin kodning så som de anser den vara bäst?
Majoriteten av respondenterna har inte tidigare arbetat enligt ett uttalat sätt att skriva sin kod på. En av respondenterna som arbetat längre med programmering har dock försökt följa riktlinjer inom andra programmeringsspråk för att skriva bättre kod.
Fråga 6. Tror du att ett uttalat gemensamt tankesätt (som grundas i Clean codes riktlinjer) inom teamet kan vara en fördel? Varför tror du det?

Respondenterna tror att ett uttalat gemensamt tankesätt, som grundats i Clean codes riktlinjer, kan vara en fördel. Det gör att alla försöker skriva kod på liknande sätt och resulterar i att alla kan hoppa in i varandras kod vid alla tillfällen.
Fråga 7. Tror du att ett uttalat gemensamt tankesätt (som grundas i Clean codes riktlinjer) inom teamet kan vara en nackdel? Varför tror du det?
En nackdel som framkommit är om det är svårt att komma överens om vilka riktlinjer som ska följas. En annan nackdel är om dessa riktlinjer tas för regler istället. Att de följs slaviskt vilket inte heller i slutändan blir bra. Att man måste följa riktlinjerna till en viss del och anpassa de efter situationen.
Fråga 8. Hur tror du att ett gemensamt tankesätt som grundar sig i Clean Codes riktlinjer kan hjälpa teamet att nå målet: skapa en lättläst och förvaltningsbar kodbas?
Respondenterna tror att ett gemensamt tankesätt, som grundas i Clean codes riktlinjer, kan hjälpa till att uppnå målet som är att försöka skapa en lättläst och förvaltningsbar kodbas. Att man använder beskrivande namn på klasser, metoder och variabler samt att detta medför att mindre eller ingen dokumentation behövs. Det är något som bidrar till att en lättläst och förvaltningsbar kodbas utvecklas. Komma överens om abstraktionsnivåer att här sker detta och där sker det.
Fråga 9. Vad kan du göra som individuell programmerare för att sprida tankesättet inom teamet?
Att dela med sig av tips man hittar eller delge information som man som individuell programmerare snappat upp är ett bra sätt. Det kan vara böcker, hemsidor eller liknande. Något som alla tar upp är att man som individuell programmerare kan tillsammans med en annan teammedlem parprogrammera för att sprida sin kunskap. Ett annat alternativ är också att mobprogrammera vilket då görs utav hela teamet.

Tabell 6. Sammanfattning av intervjuer, del 2.

4.1.3 Teamets arbete mot etablering av det gemensamma tankesättet

Idag försöker teamet tillämpa riktlinjer som betydelsefulla namn, metoder, klasser och även enhetstest i det dagliga arbetet med den gemensamma kodbasen. Svårighet med att tillämpa de riktlinjer som används idag är att ge beskrivande namn till variabler, metoder och klasser. Att de tänker att de kan komma på ett bättre namn vid ett senare tillfälle. En riktlinje som även beskrivs som svår att tillämpa är enhetstest.

De tillvägagångssätt som används för att sprida kunskapen kring Clean codes riktlinjer idag är parprogrammering och mobprogrammeing. Även någon slags dragning/föresläningar används också.

Code review, på svenska kodgranskning, beskrivs som ett sätt som är bra att använda sig av när det skrivits enklare kod enskilt.

I tabellen nedan (se tabell 7) presenteras en sammanställning av svaren på respektive intervjufråga som ingick i del tre. (Se bilaga 1)

Fråga 1. Har du idag börjat tillämpa några av de riktlinjer som Clean Code förespråkar? Vilka isåfall?
De riktlinjer som respondenterna hittils känner att de använder sig av i arbetet just nu är: - Betydelsefulla namn. Att man ser till att beskriva klasser, metoder och varibler med beskrivande namn så att man inte behöver kommentera i onödan.

- Metoder. Att inte skriva för långa metoder och att bryta ut metoder om det är en lång metod för att underlätta. Mycket av den gamla funktionaliteten används i det nya systemet och då innebär det refaktorisering när metoder bryts ut. - Enhetstest. Att man ibland använder sig av enhetstester på de ställen där detta är möjligt eftersom detta ger fortsatt hjälp när man fortsätter göra förändringar. - Klasser. Att det är bättre att klasser utför färre saker än att det blir för stora och komplexa. Boy Scout Rule är något som två respondenter tar upp, att alltid lämna koden snyggare än den var innan. Det betyder inte en radikal förändring utan kan vara en liten förändring bara.
Fråga 2. Hur anser du att det går? Ser du något ”problem” i att tillämpa Clean Code-tänket? Finns det möjligtvis några riktlinjer som du anser är svårare att tillämpa än andra?
Svårt att hålla reda på om metoder redan finns skrivna någonstans när man börjar bryta ut en stor metod till flera mindre. Kan vara svårt med betydelsefulla namn. Att det kanske tar ett tag att komma på ett bra beskrivande namn för en metod exempelvis. Att ibland tänker man att man döper om det senare men att det då riskerar att aldrig göras, later equals never. Enhetstest är också något som en respondent beskriver är svårare än andra och tar tid att vänja sig vid. Att det är svårt att få in det tänket att man först ska skriva test och sedan kod eftersom de inte arbetar med testdriven utveckling. Respondenterna anser också att när man inte är så insatt i riktlinjerna påverkar det tillämpningen.
Fråga 3. Finns det något tillvägagångssätt som ni idag som team använder er av för att sprida kunskapen kring Clean Coding?
Par- samt mobprogrammering är de som respondenterna tar upp. Även att man ibland kan göra kodgranskning eller ha någon slags dragning/föreläsning.
Fråga 4. Finns det något ytterligare tillvägagångssätt du tror skulle gynna införandet av det nya gemensamma tankesättet inom teamet? Vilket eller vilka isåfall?
Inte så många ytterligare tillvägagångssätt som respondenterna kommer på just nu. Code reviews är dock något som tas upp som skulle kunna användas för att sprida kunskapen kring Clean codes riktlinjer.
Fråga 5. Varför tycker du du att detta tillvägagångssätt kan vara en fördel för införandet av det gemensamma tankesättet som grundar sig i Clean Codes riktlinjer och principer?
Code reviews kan vara bra om man exempelvis skrivit något enklare själv. Att två hjärnor som läser koden är bättre än en. Respondenterna beskriver parprogrammering som ett tillvägagångssätt där det är lätt att sprida kunskap. Att man kanske uppfattat saker olika och kan då samtidigt diskutera detta medans man kodar vilket av respondenterna ses som den största fördelen. Respondenterna ser mobprogammering som en fördel därför att kunskap kan spridas till en större grupp än vid parprogrammering då endast två programmerare ingår samtidigt.

Tabell 7. Sammanfattning av intervjuer, del 3

4.2 Sammanställning av enkätsvar

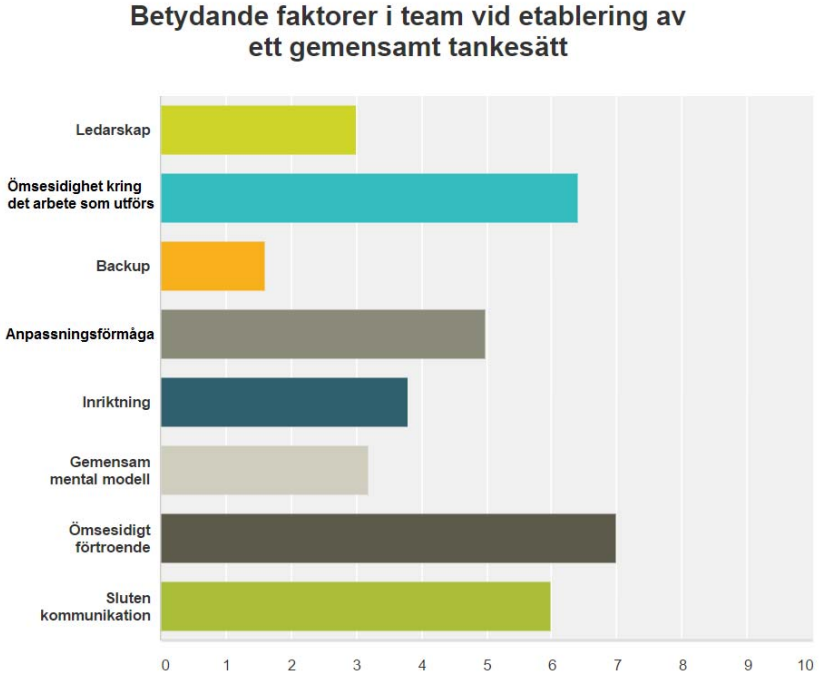
I detta avsnitt presenteras resultatet av enkätsvaren i både tabell och diagram som skapats med hjälp av enkätverktyget SurveyMonkey.

I figuren nedan (se figur 8) så syns en sammanställning av enkätsvaren i tabellform. Resultatet av enkäten visar till exempel att faktorn ledarskap rankats som fjärde viktigaste faktor två gånger, sjätte viktigaste faktor en gång och åttonde faktor två gånger.

	1	2	3	4	5	6	7	8	Totalt	Score
Ledarskap	0,00% 0	0,00% 0	0,00% 0	40,00% 2	0,00% 0	20,00% 1	0,00% 0	40,00% 2	5	3,00
Ömsesidighet kring det arbete som utförs	40,00% 2	20,00% 1	0,00% 0	20,00% 1	20,00% 1	0,00% 0	0,00% 0	0,00% 0	5	6,40
Backup	0,00% 0	0,00% 0	0,00% 0	0,00% 0	0,00% 0	0,00% 0	60,00% 3	40,00% 2	5	1,60
Anpassningsförmåga	0,00% 0	20,00% 1	20,00% 1	0,00% 0	60,00% 3	0,00% 0	0,00% 0	0,00% 0	5	5,00
Inriktning	0,00% 0	20,00% 1	0,00% 0	20,00% 1	0,00% 0	40,00% 2	0,00% 0	20,00% 1	5	3,80
Gemensam mental modell	0,00% 0	0,00% 0	20,00% 1	0,00% 0	0,00% 0	40,00% 2	40,00% 2	0,00% 0	5	3,20
Ömsesidigt förtroende	40,00% 2	20,00% 1	40,00% 2	0,00% 0	0,00% 0	0,00% 0	0,00% 0	0,00% 0	5	7,00
Sluten kommunikation	20,00% 1	20,00% 1	20,00% 1	20,00% 1	20,00% 1	0,00% 0	0,00% 0	0,00% 0	5	6,00

Figur 8. Sammanställning av enkätsvaren i tabellform.

Det är tabellens data i kolumnen ”Score” längst till höger som ligger till grund för det diagram som presenteras i figuren nedan (se figur 9).



Figur 9. Sammanställning av enkätsvar i diagram.

5. Analys

I detta kapitel presenteras de två analyser som gjorts baserat på empirin som samlats in genom intervjuer och enkäter. I analysen har insamlat data även jämförts med delar ur teorin.

5.1 Nulägesanalys

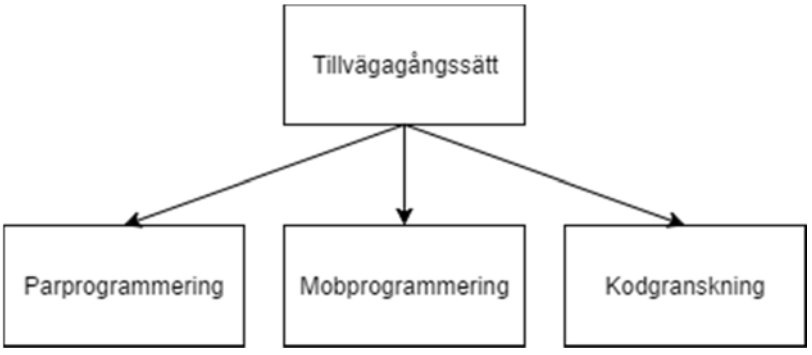
I denna del kommer nulägesanalysen att beskrivas vilken grundas på det empiriska data som samlats in genom intervjuer (4.1). Det empiriska materialet har även jämförts med delar av teoriavsnittet (kap 2). Denna del bidrar som underlag för att besvara första frågan i frågeställningen som är ”Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?”.

Tillvägagångssätt

Idag används olika tillvägagångssätt inom teamet för att etablera och sprida kunskapen kring det gemensamma tankesätt som grundas i Clean codes riktlinjer. (Se figur 10) De tillvägagångssätt som används idag är parprogrammering, mobprogrammering, kodgranskning och i enstaka fall även någon föreläsning eller dragning. Det tillvägagångssätt som används mest frekvent är i dagsläget parprogrammering.

Parprogrammering är enligt Beck (2000) ett ypperligt sätt att dela kunskap med andra vilket också bekräftas av respondenterna i teamet. Alla i teamet verkar ha en positiv bild och erfarenhet av detta tillvägagångssätt så här långt. Teamet utnyttjar i dagsläget en teammedlem med mer kunskap inom clean coding tillsammans med de som fortfarande lär sig om riktlinjerna. Genom att göra på det viset så kan kunskapsgapet som Beck (2000) pratar om fyllas och det blir ett effektivt sätt för teamet att sprida kunskapen av det gemensamma tankesättet.

Mobprogrammering är även det ett tillvägagångssätt som används. Detta sätt används mer frekvent när det kommer till större problem vilket anses vara exempelvis när större metoder ska brytas ut i flera mindre metoder. Det är ett tillvägagångssätt där alla i teamet ingår och fokuserar på att lösa ett problem. Kodgranskning där teammedlemmar manuellt granskar varandras kod är något som i dagsläget används sällan.



Figur 10. De tillvägagångssätt som används idag för att etablera det gemensamma tankesätt som grundas i Clean codes riktlinjer.

Tillämpningen av det gemensamma tankesättet

Tillämpningen av det gemensamma tankesättet, det vill säga tillämpningen av Clean codes riktlinjer, har påbörjats i teamet. Av respondenternas svar så tillämpas riktlinjer så som betydelsefulla namn (3.2.2), funktioner (3.2.1), enhetstest (3.2.5) och klasser (3.2.6) vilket betyder att totalt fyra riktlinjer utav nio tillämpas i det dagliga arbetet.

Förhållningssätt till det gemensamma tankesättet

Enligt respondenterna så är det viktigt att försöka skriva Clean code eftersom de i framtiden inte vet vem som kommer förvalta koden men även för att underlätta för sina egna teammedlemmar. Detta tyder på att det finns en positiv attityd till det gemensamma tankesättet som grundas i Clean codes riktlinjer. Utifrån detta bekräftas även är att det finns ett tydligt engagemang i teamet, vilket är en viktig del i ett gemensamt tankesätt (2.3).

Enligt Martin (2009), grundaren till begreppet Clean code, så är det viktig att dessa riktlinjer inte ses som regler, utan som riktlinjer, vilka de är. Detta är något bekräftas av respondenterna som är tydliga med att poängtera att det gemensamma tankesättet som grundas i Clean codes riktinjer, endast är riktlinjer och inte regler.

Från resultatet av intervjuerna visas även tydligt att medlemmarna i teamet är noga med att det gemensamma tankesättet endast är riktlinjer för hur de ska arbeta, inte regler precis som Martin (2009), grundaren till begreppet Clean code, också är noga med att påpeka.

5.2 Betydande faktorer i team

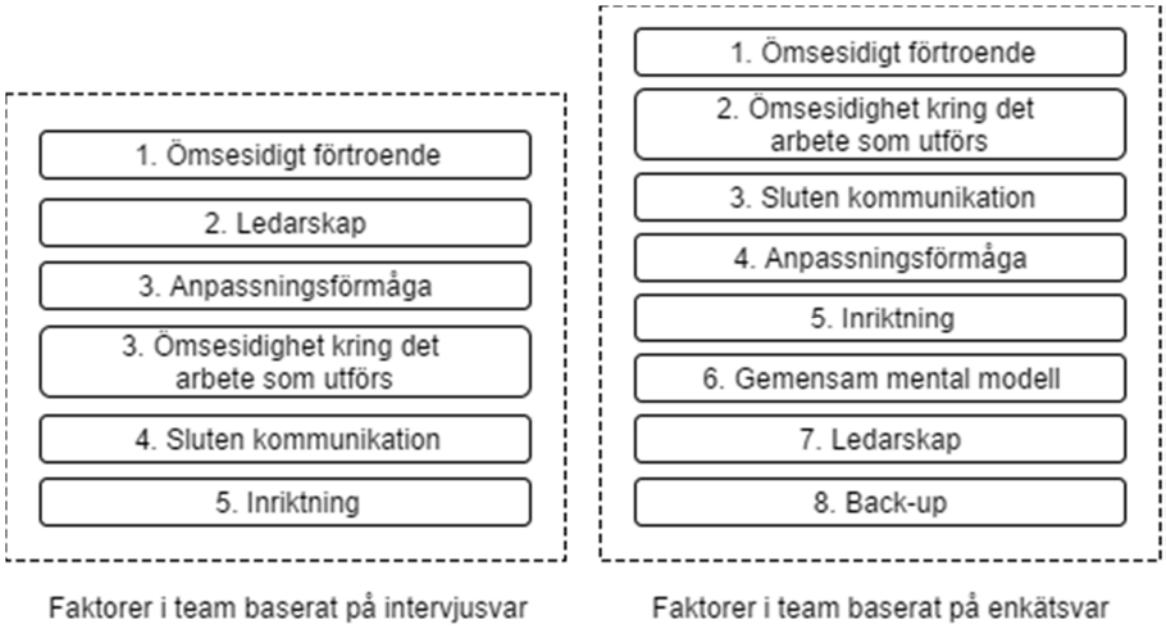
I denna del analyseras det empiriska data från både intervjuer (4.1) och enkäter (4.2) och jämförs med tidigare studier (2.10). Denna del av analysen bidrar som underlag till att besvara frågan: ”Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?”.

I figuren nedan (se figur 11) så syns tydligt att ömsesidigt förtroende är den faktor som anses vara viktigast i ett team, både när frågan angående betydande faktorer i team vinklats mer generellt (intervju del 1, se bilaga 1) och även när de vinklats till ett mer specifikt perspektiv som innebar när ett gemensamt tankesätt ska etableras (enkät, se bilaga 2). Studien som utfördes av Andreassen (2015) visade att ömsesidigt förtroende är en viktig faktor i storskaliga team, vilket här bekräftas vara en viktig faktor även i ett team som arbetar mot att etablera ett gemensamt tankesätt.

Något som skiljer sig är faktorn ledarskap. När frågan om betydande faktorer i ett team vinklats mer generellt under intervjuerna så har ledarskap setts som en av de viktigaste faktorerna i ett team. Däremot när frågan om betydande faktorer i ett team vinklats till ett mer specifikt perspektiv som innebar när ett gemensamt tankesätt ska etableras så hamnar ledarskap långt ner på listan.

Ömsesidighet kring det arbete som utförs anses vara en viktig faktor i båda fallen. Dock anses den vara lite mer betydande när frågan vinklats mer mot ett specifikt perspektiv som innebar när ett gemensamt tankesätt ska etableras. Ömsesidighet kring det arbete som utförs är en faktor som kopplas ihop med tillvägagångssätt som används av teamet på Nethouse idag då dessa tillvägagångssätt förespråkar att man tillsammans hjälps åt att skriva bättre kod.

Faktorerna anpassningsförmåga och tvåvägskommunikation skiljer sig inte markant utan är av liknande betydelse i båda fallen.



Figur 11. Jämförelse av betydande faktorer i team utifrån intervju- och enkätsvar.

6 Diskussion

I detta kapitel förs en diskussion kring de resultat som framkommit i analysen. Först diskuteras resultatet från nulägesanalysen (6.1) och sedan resultatet från analysen över betydande faktorer i team (6.2). Metodkritik tas även upp i detta kapitel (6.3).

6.1 Nulägesanalys

Manuell kodgranskning är ett tillvägagångssätt som inte används så mycket inom teamet. En anledning till detta kan möjligtvis vara att det uppfattas som ett tidkrävande moment att skicka kodsnuttar fram och tillbaks mellan varandra. Dock visar en jämförande studie att kodgranskning i systemutvecklingsprocessen kan hjälpa till att minska hälften av misstag som uppstår i koden. En annan fördel med kodgranskning är att tillvägagångssättet hjälper till att öka kodkvaliteten. (Cohen et. al, 2006) Kodgranskning skulle därför möjligtvis kunna vara ett tillvägagångssätt som hjälper till att göra kodbasen mer lättläst och förvaltningsbar om det används mer frekvent.

Endast fyra av de nio riktlinjer som Clean code förespråkar används idag vilket antas bero på att etableringen av det gemensamma tankesättet som grundas i Clean codes riktlinjer är en pågående process, något som behöver växa fram under en längre tid och arbetas in successivt.

En annan anledning skulle kunna vara att det är svårare att tillämpa vissa riktlinjer i förvaltning än vad det är i ett helt nytt projekt där man börjar från början.

En riktlinje som kommentarer (2.2.9) tror jag dock används idag även fast den inte uttalats under intervjuerna. Anledningen till detta antagande är att en riktlinje som betydelsefulla namn (2.2.2) hjälper till att minska kommentarer i koden till en viss del, vilket gör att även riktlinjen kommentater tillämpas.

En mätning om ett gemensamt tankesätt som grundas i Clean codes riktlinjer förbättrar läsbarhet och förvaltningsbarhet har inte varit möjlig att utföra i denna studie. Resultatet från en studie genomförd av Mathieu et. al (2000) tyder på att en gemensam mental modell, vilket i detta sammanhang kan antas vara det gemensamma tankesättet som grundas i Clean codes riktlinjer, har en positiv inverkan på ett teams prestationer. Resultatet från tidigare studie tillsammans med att den tydliga positiva attityden gentemot detta gemensamma tankesätt finns inom teamet så kan en förbättring av läsbarhet och förvaltningsbarhet tros kunna vara möjlig.

En anledning till att det finns ett bra förållningssätt gentemot det gemensamma tankesättet kan antas vara för att det är ett gemensamt beslut, ingen har pekat med hela handen och sagt att nu gör vi såhär. Vid ett tillfälle där det inte är ett gemensamt beslut så är det också större risk att riktlinjerna ses som regler vilka de inte ska.

6.2 Betydande faktorer i team

Resultatet från analysen tyder på att ömsesidigt förtroende är den faktor som anses vara viktigast rent generellt i ett team men även vid en specifik situation när ett gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras inom teamet. Anledningen till detta kan bero på att ömsesidigt förtroende är det som gör ett team till just ett team. Att tillsammans kunna ha förtroende att alla utför sitt arbete i teamet för att tillsammans fungera som ett team. I en situation där ett nytt gemensamt

tankesätt som grundas i Clean codes riktlinjer ska etableras så kan detta ömsesidiga förtroende antas betyda att medlemmarna i teamet tar det enskilda ansvaret att läsa på om riktlinjerna.

Ledarskap är en faktor som anses vara av mer betydelse om man vinklar frågan mer generellt för ett team än vid situationen då ett gemensamt tankesätt ska etableras inom teamet. Anledningen varför ledarskap inte alls anses vara lika betydande vid situationen då ett gemensamt tankesätt ska etableras kan antas vara för att det är just ett gemensamt tankesätt. Någonstans har teamet kommit fram till att tillsammans arbeta enligt ett uttalat sätt och då är det inte en utvald ledare som ser till att detta sker utan teamet tillsammans.

Ömsesidighet kring det arbete som utförs är en faktor som hamnar högt på listan av betydande faktorer i team när ett nytt gemensamt tankesätt ska etableras. Anledningen till detta resultat kan vara för att det är viktigt att våga lägga sig i andra teammedlemmars kod för att kunna sprida kunskapen kring det gemensamma tankesättet. Eller också tvärt om, att låta någon lägga sig i för att sprida sin kunskap.

Ömsesidighet kring det arbete som utförs är en faktor som kan kopplas ihop med de tillvägagångssätt som används av teamet idag. Anledningen till varför jag ser en koppling är för att faktorn ömsesidighet kring det arbete som utförs innebär att hitta ett sätt där feedback kan ges och även där misstag kan identifieras utan att ses som något dåligt. Tillvägagångssätten som används verkar vara ett svar på hur denna faktor kan beaktas.

Tvåvägskommunikation är en faktor som anses vara viktig även den. Anledningen till detta kan vara att det i en situation där ett nytt gemensamt tankesätt som handlar om hur kod ska skrivas är viktigt med en tvåvägskommunikation. Att teammedlemmarna vågar fråga om det är någon riktlinjer de tros ha missuppfattat eller om det är någonting i kosbasen som är oklart. Tvåvägskommunikation är även den en faktor som kan kopplas ihop med de tillvägagångssätt som används av teamet idag. Finns ingen Tvåvägskommunikation inom teamet när tillvägagångssätten används så är risken stor att de inte kommer fungera heller.

6.3 Metodkritik

I början av undersökningen är det svårt att veta om de metoder som valt ut att användas för att samla in data är de som i slutändan kommer hjälpa studien uppfylla syftet och generera ett bra resultat. Både intervjuer och enkäter användes i denna studie för ha möjlighet att uppnå syftet.

Intervjuerna skedde väldigt tätt inpå varandra vilket på sätt och vis var bra, men vilket även gjorde att jag inte fick någon tid mellan intervjuerna till att reflektera över om de intervjufrågor jag valt att använda gav de svar jag var ute efter. Jag kan nu i efterhand se att det hade vart bättre om frågan gällande betydande faktorer inte vinklats så generellt utan om den hade vinklats mer mot ett team i en situation där ett gemensamt tankesätt ska etableras. Detta hade gett en bättre validering och jämförelse av faktorerna mellan intervju- och enkätsvar.

Två enkäter utformades och användes. När den första enkäten analyserades så var det svårt att dra några slutsatser kring svaren och se något samband. I den första enkäten användes svar i form av en skala från 1-5 vilket gjorde det svårt för mig att dra några slutsatser. Svarsalternativen hade kunnat bytas ut mot en annan typ av svarsalternativ. Därför utformades en ny enkät som ansågs ge ett bättre resultat. Eftersom detta moment skedde i slutet av perioden så hade jag inte riktigt tid att fundera över ytterligare frågor som hade kunnat vara bra att ha med i den nya enkäten vilket kan ha påverkat resultatet på studien. Resultatet från enkäten som valdes att uteslutas från studien finns att tillgå som separat dokument.

7 Slutsats

I detta kapitel så presenteras svaren på de frågeställningar som använts för att uppnå studiens syfte. (7.1 & 7.2) Kunskapsbidraget presenteras under ett eget avsnitt (7.3) och även förslag på framtida studier (7.4).

Syftet med studien var att beskriva hur ett team går tillväga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer. Studien syftade även till att beskriva betydande faktorer som anses vara viktiga att beakta när ett gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras.

För att ha möjlighet att uppfylla studiens syfte så formulerades två forskningsfrågor:

- Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?
- Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

Svar på dessa forskningsfrågor ges under kommande två avsnitt.

7.1 Hur arbetar teamet med etableringen av det gemensamma tankesättet idag?

Utifrån nulägesanalysen så kan det konstateras att teamet idag arbetar med att etablera det gemensamma tankesättet, som grundas i Clean codes riktlinjer, genom att använda tillvägagångssätt som:

- Parprogrammering
- Mobprogrammering
- I enstaka fall manuell kodgranskning

Dessa tillvägagångssätt förespråkar att teammedlemmarna arbetar i par eller grupp och tillsammans skriver kod. Genom att arbeta på det här sättet så så sprids kunskapen kring Clean codes riktlinjer på ett smart sätt i teamet eftersom de utnyttjar varandras kunskaper.

7.2 Vilka faktorer kan anses som viktiga att beakta när ett nytt gemensamt tankesätt ska etableras?

Utifrån resultatet av analysen så drar jag slutsatsen om att det finns fyra faktorer som är av större betydelse och bör beaktas. Tre av dessa faktorer gäller teamwork och är:

- **Ömsesidigt förtroende** – Det ömsesidiga förtroendet är viktigt för att varje enskild utvecklare tar ansvar för att lära sig om Clean codes riktlinjer och vad de innebär och att sedan också i den mån det går tillämpa dessa i sitt dagliga arbete.

- **Tvåvägskommunikation** – Det är viktigt att det finns en tvåvägskommunikation inom teamet där det känns tillåtet att fråga om man inte förstår. Det gäller även att den som förklarar något för någon har ansvar att kontrollera så att informationen uppfattats som det var tänkt. Speciellt i sammanhanget där riktlinjer kring hur kod ska skrivas tillämpas.

- **Ömsesidighet kring det arbete som utförs** – En ömsesidighet kring det arbete som utförs är viktig i denna situation för att det gemensamma tankesätt som ska etableras blir korrekt. Att man kan lägga sig i andras kod och tvärt om för att se till att dessa riktlinjer tillämpas korrekt och i den mån det går.

Den fjärde faktorn är inte en uttalad faktor gällande teamwork men är en betydande faktor som även bör beaktas. Särskilt när ett nytt gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras.

- **Tillvägagångssätt** – Tillvägagångssättet är kugghjulet som för ihop teamet, det gemensamma tankesättet och de betydande faktorerna gällande teamwork på ett effektivt sätt. Parprogrammering är ett bra exempel. I ett tillvägagångssätt som parprogrammering förstärks viktiga faktorer inom teamwork så som tvåvägskommunikation och ömsesidighet kring det arbete som utförs.

7.3 Kunskapsbidrag

Ett av kunskapsbidragen som genererats utifrån studien har varit att beskriva hur teamet på Nethouse arbetar med att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer. Ett annat kunskapbidrag som studien genererat är faktorer som anses vara viktiga att beakta när ett nytt gemensamt tankesätt som grundas i Clean codes riktlinjer ska etableras.

Kunskapsbidragen kan vara av intresse för team som någon gång funderat på att skriva kod enligt Clean codes riktlinjer. Kunskapsbidragen kan även vara av intresse för andra team som arbetar med förvaltning samt för team som står inför nyutveckling. Resultatet kan även vara av intresse rent generellt då den bidrar med nyttig kunskap kring viktiga faktorer som överlag är värda att beakta vid arbete med utveckling och förvaltning.

7.4 Förslag på framtida studier

Förslag på en framtida studie är att undersöka om det finns team på andra företag som använder sig av Clean codes riktlinjer i sitt arbete gällande utveckling och förvaltning och hur dem gått tillväga för att etablera dessa riktlinjer inom teamet.

Ett annat förslag på framtida studie är att, om det finns andra team som arbetar enligt Clean codes riktlinjer, göra en jämförelse med denna studie för att se om det finns några likheter eller skillnader i hur dessa riktlinjer etableras i ett team.

Det kändes svårt att ta fram rekommendationer för hur ett team ska gå till väga för att etablera ett gemensamt tankesätt som grundas i Clean codes riktlinjer genom att basera dessa rekommendationer på ett enstaka fall. Förslag på en ytterligare framtida studie är att, om det finns fler fall av denna typ att studera, använda denna studie som underlag tillsammans med ytterligare fall för att ta fram rekommendationer för detta.

Referenser

- Agile Alliance. (2015). *Mob Programming – A Whole Team Approach.* Hämtad 2017-04-21, från https://www.agilealliance.org/wp-content/uploads/2015/12/ExperienceReport.2014.Zuill_.pdf
- Andreassen, E. (2015). Inter-team Coordination in Large-scale Agile Software Development-An Exploratory Case Study (Master's thesis, NTNU).
- Beck, Kent. Extreme programming explained: embrace change. addison-wesley professional, 2000.
- Björklund, M. & Paulsson, U. (2012). Seminarieboken: att skriva, presentera och opponera. (2. uppl.) Lund: Studentlitteratur.
- Code Review. (2017, 28 april). I *Wikipedia.* Hämtad 2017-05-07, https://en.wikipedia.org/wiki/Code_review
- Cohen, J., Brown, E., DuRette, B., & Teleki, S. (2006). Best kept secrets of peer code review. Smart Bear.
- Dingsøyr, T., & Dybå, T. (2012, June). Team effectiveness in software development: Human and cooperative aspects in team effectiveness models and priorities for future studies. In Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering (pp. 27-29). IEEE Press.
- Dyer, J. L. (1984). Team research and team training: A state-of-the-art review. In F. A. Muckler, A. S. Neal, & L. Strother (Eds.), Human factors review (pp. 285-323). Santa Monica, CA: Human Factors Society
- Engagemang. (2015, 11 juli). I *Wiktionary*. Hämtad 2017-04-20, https://sv.wiktionary.org/wiki/engagemang
- Fowler, M., & Beck, K. (1999). Refactoring: improving the design of existing code. Addison-Wesley Professional.
- Funktion (programmering). (2017, 11 februari). I *Wikipedia*. Hämtad 2017-04-10, https://sv.wikipedia.org/wiki/Funktion_(programmering)
- Haverblad, A. (2007). IT Service Management i praktiken. Studentlitteratur.
- Heitlager, I., Kuipers, T., & Visser, J. (2007, September). A practical model for measuring maintainability. In Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the (pp. 30-39). IEEE.
- Lerthathairat, P., & Prompoon, N. (2011, May). An approach for source code classification to enhance maintainability. In Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on (pp. 319-324). IEEE.
- Martin, R. C. (2009). Clean code: a handbook of agile software craftsmanship. Pearson Education.
- Mathieu, J. E., Heffner, T. S., Goodwin, G. F., Salas, E., & Cannon-Bowers, J. A. (2000). The influence of shared mental models on team process and performance. Journal of applied psychology, 85(2), 273.
- Mindset. (2017, 18 april). I *Wikipedia.* Hämtad 2017-04-20, https://en.wikipedia.org/wiki/Mindset
- Nationalencyklopedin, arbetsgrupp. http://www.ne.se/uppslagsverk/encyklopedi/lång/arbetsgrupp (hämtad 2017-04-11)
- Non-functional requirements. (2017, 26 mars). I *Wikipedia*. Hämtad 2017-04-11, https://en.wikipedia.org/wiki/Non-functional_requirement
- Nulägesanalys. (2015, 27 maj). I *Wikipedia.* Hämtad 2017-05-11, https://sv.wikipedia.org/wiki/Nul%C3%A4gesanalys
- Oates, B.J. (2006). Researching information systems and computing. London: SAGE.
- Rosene, A. F., Connolly, J. E., & Bracy, K. M. (1981). Software maintainability-what it means and how to achieve it. IEEE Transactions on Reliability, 30(3), 240-245.

- Salas, E., Dickinson, T.L., Converse, S.A., & Tannenbaum, S.I. (1992). Toward an understanding of team performance and training. In R. W. Swezey & E. Salas (Eds.), Teams: Their training and performance (pp. 3-29). Norwood, NJ: Ablex.
- Salas, E., Sims, D. E., & Burke, C. S. (2005). Is there a “big five” in teamwork?. Small group research, 36(5), 555-599.
- Srinivasulu, D. (2012). Evaluation of Software Understandability Using Software Metrics.
- Systems development life cycle. (21 maj, 2017) I *Wikipedia*. Hämtad 2017-05-22, https://en.wikipedia.org/wiki/Systems_development_life_cycle
- Ward Cunningham. (2016, 23 oktober). I *Wikipedia*. Hämtad 2017-04-10, https://sv.wikipedia.org/wiki/Ward_Cunningham
- Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. MIS quarterly, xiii-xxiii.

Bilagor

Bilaga 1 – Intervjufrågor

Del 1

1. Hur länge har du jobbat med programmering?
2. Hur länge har du jobbat i teamet på Nethouse?
3. Vilka är dina uppfattningar av att jobba i team? Positiva och negativa.

Det finns 8 uttalade faktorer som anses vara viktiga för att ett team ska fungera bra och dessa är:

- **Ledarskap** – att det finns någon person som ses som mer styrande i teamet.
- **Ömsesidighet kring det arbete som utförs** - med detta menas att det är högt i tak och att det känns okej att lägga sig i andra teammedlemmars kod och även tvärt om.
- **Backup** - att det finns en plan för hur teamet går till väga för att klara tungt belastade perioder.
- **Anpassningsförmåga** - hur lätt teamet och de individer som ingår har för att anpassa sig till nya tankesätt och arbetssätt.
- **Inriktning** - en inriktning med tydligt mål finns definierat inom teamet.
- **Gemensam mental modell** – kan ses som en modell som bidrar med kunskap i hur teamet ska arbeta och hur medlemmarna ska interagera med varandra.
- **Ömsesidigt förtroende** - handlar om att medlemmar i teamet ska dela uppfattningen av att varje medlem tar ansvar över sin roll i teamet och det arbete som ska utföras.
- **Tvåvägskommunikation** - innebär att det finns en kommunikation mellan teammedlemmarna och att man följer upp för att se att informationen verkligen gått fram.

4. Vilka tre av dessa åtta faktorer anser du är viktigast för ett team? Varför?
5. Vad anser du är viktigt när det kommer till team och programmering i kombination med varandra?
6. Tycker du att det finns någon problematik att som programmerare koda i ett team? Varför?

Del 2

1. Har du tidigare hört talas om begreppet Clean Code och vet vad det omfattar?
2. Har du tidigare tänkt på hur du skriver din kod?
3. Tycker du att det är viktigt att skriva Clean Code?
4. Varför tycker du att det är viktigt att skriva Clean Code?
5. Har du någonsin tidigare jobbat enligt ett utalat sätt att skriva din kod på? Eller har det mer varit att var och en sköter sin kodning så som de anser den vara bäst?
6. Tror du att ett uttalat gemensamt tankesätt (som grundas i Clean codes riktlinjer) inom teamet kan vara en fördel? Varför tror du det?
7. Tror du att ett uttalat gemensamt tankesätt (som grundas i Clean codes riktlinjer) inom teamet kan vara en nackdel? Varför tror du det?
8. Hur tror du att ett gemensamt tankesätt som grundar sig i Clean Codes riktlinjer kan hjälpa teamet att nå målet: skapa en lättläst och förvaltningsbar kodbas?
9. Vad kan du göra som individuell programmerare för att sprida tankesättet inom teamet?

Del 3

1. Har du idag börjat tillämpa några av de riktlinjer som Clean Code förespråkar? Vilka isåfall?
2. Hur anser du att det går? Ser du något ”problem” i att tillämpa Clean Code-tänket? Finns det möjligtvis några riktlinjer som du anser är svårare att tillämpa än andra?
3. Finns det något tillvägagångssätt som ni idag som team använder er av för att sprida kunskapen kring Clean Coding?
4. Finns det något ytterligare tillvägagångssätt du tror skulle gynna införandet av det nya gemensamma tankesättet inom teamet? Vilket eller vilka isåfall?
5. Varför tycker du du att detta tillvägagångssätt kan vara en fördel för införandet av det gemensamma tankesättet som grundar sig i Clean Codes riktlinjer och principer?

Bilaga 2 – Enkät

Betydande faktorer i team vid etablering av ett gemensamt tankesätt.

När ett nytt gemensamt tankesätt ska etableras kan vissa faktorer i ett team spela större roll än andra. Om du tänker på situationen när ett gemensamt tankesätt som grundas i clean codes riktlinjer ska etableras i ett team, vilka faktorer anser du är viktigare än andra i en sådan situation?

1. Rangordna faktorerna nedan från 1-8 genom att antingen dra i respektive faktor upp och ned eller genom att välja ett nummer i listan till vänster om respektive faktor.

(1 mycket viktig, 8 mindre viktig)

- **Ledarskap** – att det finns någon person som ses som mer styrande i teamet.
- **Ömsesidighet kring det arbete som utförs** - med detta menas att det är högt i tak och att det känns okej att lägga sig i andra teammedlemmars kod och även tvärt om.
- **Backup** - att det finns en plan för hur teamet går till väga för att klara tungt belastade perioder.
- **Anpassningsförmåga** - hur lätt teamet och de individer som ingår har för att anpassa sig till nya tankesätt och arbetssätt.
- **Inriktning** - med inriktning menas att det t.ex. finns ett tydligt mål definierat kring vad man vill uppnå med arbetet inom teamet.
- **Gemensam mental modell** – kan ses som en modell som bidrar med kunskap i hur teamet ska arbeta och hur medlemmarna ska interagera med varandra.
- **Ömsesidigt förtroende** - handlar om att medlemmar i teamet ska dela uppfattningen av att varje medlem tar ansvar över sin roll i teamet och det arbete som ska utföras.
- **Sluten kommunikation** - Att det finns en tvåvägskommunikation mellan medlemmar i teamet som innebär att man vågar fråga om det är något som man inte förstår.

Bilaga 3 – Transkriberade intervjuer

Intervju med respondent 1
2017-04-12

Jag: Hur länge har du jobbat med programmering?

R1: Ehh, ja. Jobbat med det, det är i två år snart. Fast inte på heltid utan jag har även varit scrum master som har tagit.. Så effektivt kanske knappt ett år.

Jag: Okej. Hur länge har du jobbat i teamet på Nethouse?

R1: I två år.

Jag: Okej. Vilka är dina uppfattningar av att jobba i team? Kan du säga någonting positivt och någonting negativt?

R1: Ja de är väl att man aldrig är själv. Man får alltid hjälp nära till hands. Och man lär sig av varandra. Jaa.. Man har kul.

Jag: Det är inget som känns som något negativt? Mest bara positivt?

R1: Jaa.

Jag: Det är okej om du inte kommer på något negativt också såklart.

R1: Nej, men jag måste nog fundera. Ställ frågan igen.

Jag: Vilka är dina uppfattningar av att jobba i team?

R1: Neej, jag tror inte att jag tycker någonting är negativt. Ehm, det enda skulle vara de att ibland när man ska fokusera när man sitter själv och ska programmera och ska gå in i, nej men man ska ju inte gå in i sin egen värld egentligen, i den där bubblan och tänka själv bara egentligen. Men ibland så gör man ju de för att vissa saker är ju så små så att man gör dem. Man parprogrammerar eller man mobprogrammerar inte utan man gör dem själv. Och då kan jag tycka att det är skönt att liksom inte ha så mycket störningsmoment runt omkring, men då sätter jag på mig hörlurarna istället. Så då kan jag ju liksom lösa den biten. Men annars så kommer jag inte på någonting jag tycker att, som jag ser till det här teamet, så kan jag inte komma på någonting negativt.

Jag: Det kanske är svårt också att jämföra om du inte tidigare jobbat i ett annat team?

R1: Ja precis, jag har inte sådär jättemycket att jämföra med. Man jobbade ju lite grann i grupper i skolan men då jobbade man ju oftast med den som man fungerade bra ihop med som man fick välja själv och då blev det oftast inga stora konflikter eller något sånt då. Men annars så jobbar man i team så finns det ju, ibland så finns det ju risk för att många vill väldigt mycket olika. Jag har inte upplevt det, än.

Jag: Okej, men vad bra ändå.

R1: Jaa!

Jag: Nu kommer det komma en ganska så stor fråga. Säg gärna till om du vill att jag ska upprepa mig. Det finns åtta uttalade faktorer som anses vara viktiga för att ett team ska fungera bra. **Läser upp dessa åtta uttalade faktorer** Vilka tre av dessa åtta faktorer anser du är viktiga för ditt team och varför?

R1: Jaa, okej.

Jag: Och då var det ledarskap...

R1: Jaa, det kan jag ju redan nu säga att det tycker jag ju är jätteviktigt. Fast, det är iallafall i början av ett team. När man är i startfasen innan man har kommit förbi faserna där alla roller är tydligt utspecade och ledarskap är väldigt viktigt då för att se till att teamet går framåt och att konflikter löser sig så att det kommer upp till ytan liksom. Så det tycker jag är jätteviktigt, ledarskap.

Jag: Nästa faktor var ömsesidighet kring de arbete som utförs, back-up, anpassningsförmåga, inriktning. Inriktning var tydligt gemensamt mål. Gemensam mental modell, ömsesidigt förtroende och tvåvägskommunikation.

R1: Och tvåvägskommunikation, var det att man kunde kommunicera med varandra?

Jag: Jaa, man vågar ha en kommunikation, där man man vågar fråga om det är något man missat eller..

R1: Ja, men då tycker jag också att den är väldigt viktig. Och sen är det att... får jag kolla på dem?

Jag: Jaa, gud ja! Jag tänkte att det kanske skulle bli lite svårt att komma ihåg alla. De här är de. *Visar de åtta faktorerna*

R1: Ömsesidigt förtroende skulle jag ta också.

Jag: Så tvåvägskommunikation, ömsesidigt förtroende och ledarskap.

R1: Jaa!

Jag: Då går vi vidare...

R1: Jag tycker alla är viktiga men..

Jag: Jaa, men precis men jag tänkte att ni skulle få välja ut tre av dessa för att det skulle bli lite svårare. Då går vi vidare till nästa fråga. Vad anser du när det kommer till team och programmering i kombination med varandra?

R1: Jaa. Att man har ett gemensamt synsätt. Om man jobbar med, i samma team med samma kodbas liksom. Så kan det nog kanske vara bra att man har ett någolunda gemensamt synsätt på hur man ska koda. Det tycker jag. Ehh, ställ frågan igen.

Jag: Vad anser du är viktigt när det kommer till team och programmering i kombination med varandra?

R1: Mmm, ja men sen öppenhet, att man pratar med varandra. Att man tar hjälp när man fastnar och att man inte bara försöker göra allt själv utan hellre att man tar hjälp av varandra och försöker lösa problemen tillsammans kanske. Mmm, typ.

Jag: Vi går vidare.. Säg till bara om det är så att du känner att du inte har nått mer att säga så...

R1: Jaa.

Jag: Tycker du att det finns någon problematik att som programmerare koda i ett team? Att du kanske tycker det kan vara svårt eller annat?

R1: Alltså problemet är väl om man är på väldigt olika nivåer, kunskapsmässigt, att den som är väldigt duktig kontra den som är kanske inte alls hållt på med det så länge att det blir ett väldigt stort glapp där imellan, att man pratar nästan lite olika språk. Ehh, men det är återigen kommunikation och att man delar med sig och att man hjälper saker eller att man löser saker tillsammans. Då sprider man ju kunskapen.

Jag: Aah, men vad bra. Ska vi gå vidare då?

R1: Jaa, vad svarade jag nu då..

Jag: Jag tycker att du gett jättebra svar hittils! Då tänkte jag att vi går in mer på den andra delen i intervjun och kommer handla med om clean code och gemensamt tankesätt. Och vi börjar med frågan har du tidigare hört talas om begreppet clean code och vet vad det omfattar?

R1: Jaa, jag har läst, jag har inte läst klart men jag har läst en bok om det. Uncle Bob.

Jag: Känner du igen det här då kanske? *Visar upp en modell av respektive riktlinjer* Nu blev det kanske lite liten text på bilden men det är väl ungefär som det innefattar.

R1: Jaa, men det är väl det där som är de olika kapitlena va?

Jag: Jaa, men precis.

R1: Jag har inte läst igenom allt, men ja. Jag är på väg. Drygt halvvägs har jag kommit, det tar rätt så lång tid för mig att läsa en bok för jag skriver oftast ner så jag har ett anteckningsblock med det som jag tycker är viktigast. Så jaa.

Jag: Ja, ehh. Har du tidigare tänkt på hur du skriver din kod, innan du liksom börjat tänka enligt..

R1: Tidigare har jag inte tänkt så djupt som, som jag har börjat gjort nu när jag läst den här boken. Eh, tidigare så var det väldigt, jaa hellre att det skulle gå lite snabbt och att det inte var så viktigt egentligen vad variabeln hette, typ. Bara att det typ...

Jag: Det funkade helt enkelt?

R1: Ja, men precis.

Jag: Okej, men tycker du att det är viktigt då att skriva clean code?

R1: Ehh, nu när jag sitter i det här förvaltningsprojektet, tidigare tyckte jag inte det. Då hade jag inte ens tänkt den tanken. Men nu när jag sitter i förvaltningsprojektet och ser hur pass svårt det är att förvalta en kod som är skriven av massa olika personer som skriver på olika sätt och gör som de gör på olika sätt så, jag tycker att man, clean coding är ett bra sätt att kanske komma till rätta med vissa problem iallafall. Det är ju mycket gamla kommentarer som aldrig blir uppdaterade och metodnamn och variabelnamn som man bara, vad gör det här egentligen? Metoder som är flera tusen rader långa.

Jag: Oh, ja.. Det här svarar väl kanske lite på varför du tycker att det är viktigt att skriva clean code? Som var nästa fråga. Men att det löser mycket...

R1: Jaa, men precis. Lite ordning och reda.

Jag: Mmm, har du någonsin jobbat enligt ett uttalat sätt att skriva din kod på, eller har det mer varit att var och en sköter sin kodning så som de anser den vara bäst typ?

R1: Jaa, nej jag har ju bara, jag har ju inte jobbat i något annat projekt eller team tidigare och kommer från skolan och där var det ju liksom, där tänkte man ju inte riktigt på vad man döpte variabler och metoderna till mer än att, ja den kanske kan heta något sånt här, för att det är något sånt som delvis den här metoden gör, så att jag har väl liksom inte tänkt.. Ställ frågan igen så får jag se om jag...

Jag: Har du någonsin tidigare jobbat enligt ett uttalat sätt att skriva din kod på?

R1: Jaa, så nej, jag har ju inte det. Nej.

Jag: Okej. Tror du att ett uttalat gemensamt tankesätt, som nu i detta tillfället då grundar sig i clean codes riktlinjer inom teamet kan vara en fördel?

R1: Jaa.

Jag: Varför tror du det?

R1: För att jag tror att vi kommer kunna hoppa in och jobba med varandras kod, kod som jag skrivit kan andra teammedlemmar enkelt förstå och arbeta vidare med och vise versa.

Jag: Mmm, tror du att, nu ställer jag samma fråga igen men tror du att det kan vara en nackdel med ett gemensamt tankesätt för hur man ska skriva koden?

R1: Nej, men nackdelen kan väl vara om man inte kommer överens att det finns, att man tycker olika och man inte kommer överens och man kan inte liksom bestämma hur man ska göra i vissa lägen liksom. Bara för att man, för att vi säger att vi ska ha vissa riktlinjer då kanske det, då finns det kanske flera olika förslag på någon del, såhär bör vi göra men så här behöver vi göra. Och så kommer man inte överens istället så då kanske det blir såhära irritation istället. Men jag tror att fördelarna väger upp nackdelarna. Det är väl typ det enda jag kan komma på som är nackdelar, att inte komma överens. Men då har man ju egentligen Uncle Bob eller mycket böcker att luta sig tillbaks till.

Jag: Mm, att man kan peka på det.

R1: Jaa, men precis.

Jag: Mm, hur tror du att ett gemensamt tankesätt, som grundar sig i clean codes riktlinjer, kan hjälpa teamet att nå målet att skapa en lättläst och förvaltningsbar kodbas?

R1: Hur clean coding kan hjälpa oss att uppnå det?

Jag: Jaa, hur tror du att ett gemensamt tankesätt som då grundar sig i det....

R1: Jaa, det är ju att det, att vi kommer skriva någorlunda lika och att göra på samma sätt så att inte systemet eller systemen är byggda eller skriva på massa olika sätt och man använder olika, ehm, jaa, jag tror iallafall att har man, det behöver ju inte vara clean coding men just att man har ett gemensamt tankesätt att vi inte ska använda svengelska utan vi kanske ska döpa variabler och metoder på engelska och att man har en viss, ett visst mönster, designmönster, nej inte designmönster men att, men gud nu kommer jag inte på...

Jag: Menar du hur ni skriver koden...?

R1: Abstraktionsnivåer. Att i affärslogiken, att man har, att man även kommer överens om att här bör den här koden ligga som ska hämta ut data från databasen, alltså att man har, ja, överenskomna abstraktionsnivåer liksom mellan olika skikt i koden. Från gränssnitt till databasen och upp liksom. Det kanske man också kan komma överens om.

Jag: Mmm. Då går vi vidare till nästa fråga. Vad kan du göra som individuell programmerare i teamet för att sprida tankesättet?

R1: Ehm, parprogrammera. Mobprogrammera. Har man läst någonting bra som man tycker att det här kan vara bra att resten av teammedlemmarna också vet om eller tipsar om, då kan man ju liksom bara sätta upp ett litet kort möte och dra det man vet eller fått nys om.

Jag: Ja, och då var den andra delen slut. Så vi går in på den tredje och det kommer handla lite mer om hur ni jobbar idag. Har du idag börjat tillämpa några av de riktlinjer som clean code förespråkar och vilka isåfall?

R1: Ehm, jag tänker väl mer på att variabler och metoder bör ha självbeskrivande namn. Att de verkligen beskriver vad de används till eller ja, ehm, och sen så tänker jag på att metoder inte ska vara för långa att de helst och endast ska handla om, eller göra en sak. Ehm, och ser man metoder som är väldigt väldigt långa i förvaltningen så försöker man kanske bryta ut och göra, se om man kan bryta ut

kodrader till en egen metod och så. Jag har inte kommit så jättelångt i det arbetet men det liksom så här första steg.

Jag: Ja, men precis man måste ju börja någonstans.

R1: Ja precis.

Jag: Hur tycker du att det går? Är det svårt att tänka enligt de här riktlinjerna som du har börjat med, finns det några du tror kan vara svårare att tillämpa också?

R1: Jaa, jag har redan märkt att sitta och fundera på metodnamn och variabelnamn tar lite längre tid. Men jag tror eller är rätt så säker på att om jag tänker såhär att, ja men jag döper den här metoden eller variabeln till någonting just nu, jag vet ungefär vad det är, men så tar jag och funderar lite djupare på det sen så, alltså later equals never, typ. Jaa, men det är ju så och det är ju typ precis så som det står i den här boken också och det stämmer så väl. Så att jaa..

Jag: Jaa, men de ett bra svar tycker jag. Finns det något arbetssätt idag som ni använder er av för att sprida kunskapen kring clean coding?

R1: Jaa, vi parprogrammerar och mobprogrammerar och det är delvis med en i teamet då som är väldigt duktig på det. Men vi försöker att göra saker tillsammans så att alla försöker få, så att vi får ett litet gemensamt synsätt på och försöker sprida kunskapen i teamet. Det var någon månad sen som vi satt tre stycken och bröt ut, refaktoriserade, en jättestor metod, den kanske var 600 rader någonting ner till hundra-ish nånting rader, det var typ det vi kunde göra då och då satt vi tre stycken och mobprogrammerade så att vi satt 10 minuter var ungefär och så böt vi förare liksom. Och så tycker jag, det är rätt så bra att sprida kunskap så.

Jag: Finns det något annat än mob- och parprogrammering, något annat arbetssätt på det viset som du tror skulle kunna vara liknande, eller att det skulle gynna införandet av det här gemensamma tankesättet?

R1: Jaa, det är väl att vi kanske läser, att alla läser någon bok om clean coding. Helst så ska det vara kanske samma, eller ja, några böcker som är lika. Men vi har ju en bok och det är den som jag lånat.

Jag: Är det Uncle Bobs bok?

R1: Jaa, precis och den har vi ju här i teamet så den kommer ju få gå runt...

Jag: Okej, jaa men då ska vi ta den sista frågan. Och varför tycker du att dem arbetssätten som ni jobbar med idag kan vara en fördel, eller är en fördel för införandet av det gemensamma tankesättet?

R1: Nu förstod jag inte frågan riktigt...

Jag: Varför tycker du att de här sätten, mob och parprogrammering är... vet inte om jag ska kalla det arbetssätt eller.. Men varför tycker du att dessa sätt är en fördel för införandet av det gemensamma tankesättet?

R1: Jaa. För att man enkelt sprider kunskap på det viset. Det är typ de.

Intervju med respondent 2
2017-04-12

Jag: Hur länge har du jobbat med programmering/systemutveckling, eller ja nu har ni ju mer kanske jobbat med förvaltning, men programmering i sig?

R2: Jaa, i arbetslivet då?

Jag: Jaa, men precis.

R2: Då är det, ja vad blir det nu då, det är inte så länge, det är 1,5 år ungefär.

Jag: Hur länge har du jobbat i teamet på Nethouse?

R2: Ja, det är också 1,5 år då.

Jag: Vilka är dina uppfattningar av att jobba i team? Det kan vara både positiva och negativa.

R2: Nu har ju jag jobbat i team i andra konstellationer på andra företag tidigare också. Men liksom jag upplever ju att det som är, med dem teamen jag har vart i så har jag har bra team som har fungerat bra och man får ju en gemenskap som kan saknas om man säger att man jobbar själv eller nått när man sitter på i en sån här egna kontorsrum och inte liksom jobbar nära varandra dagligen då tappar man ju lite av den, den här kontinuerliga kontakten, de sociala kopplingarna. Och sen har man ju lätt att kunna liksom prata med varandra om det är någonting som man tex kör fast på och så lite olika sånna saker. Sen negativt ibland det kan väl vara att det finns konflikter, det finns det ju i alla team och dyker sånna upp ja då kan det väl vara lite jobbigt en period, men det är ju det goda och det onda med det. När det fungerar så fungerar det bra och lite sånt. Nja, så det är väl egentligen, jag har ju haft en bra, positiv upplevelse av de team som jag jobbat i. Och de jag jobbat tidigare i, då har man ju liksom känt det är ju inte lika, man tappar ju lite av att sitta helt ensam någonstans, sen så de är min uppfattning, jag gillar att jobba i team.

Jag: Ja, men det låter ju bra. Då fortsätter vi till nästa fråga. Det finns åtta uttalade faktorer som anses vara viktiga för att ett team ska fungera bra och det är *Läser upp de åtta faktorerna* Då ska du få välja tre utav dessa som du tycker är viktigast för ditt team och anledningen till varför du väljer dessa.

R2: Ja, juste. Olika saker är ju olika viktiga vid olika tillfällen.

Jag: Jaa, det är ju viktiga saker allihopa.

R2: Jaa precis. Men en av de viktiga som är kontinuerligt det är ju ömsesidigt förtroende. Det är ju så att man vet att den ja att alla vet sina roller och det är ju liksom en del i att det bara flyter på kan man säga i teamet. Så den är ju viktig. Ehm, är det här mer specifikt för vårt team då eller?

Jag: Jaa, alltså det du tycker är viktigt i teamet du jobbar i och så...

R2: Mm, för ja då är ju, för nyare team, som med nya medlemmar och sånt då är ju ledarskap också rätt viktig i början när man har de olika faserna i teamet om man säger när man är på de första grundfaserna då krävs det ju lite ledarskap som kan fatta vissa beslut och styra lite. Det är ju inte alltid alla vet riktigt, rollerna kanske inte satt sig och sånna här saker. För sen tappar den ju sin funktion lite, på de här lite högre nivåerna när det är, om man kollar på de fyra nivåerna i effektiva team tex, då är det ju på nivån 3 och 4 då är ju teamet självgående, då behövs det inte någon stor ledarroll för alla har ju, det här ömsesidiga förtroendet liksom. När alla vet vad dem ska göra för något. Men ledarskap är ju viktigt iallafall i de här tidigare, så just nu skulle jag vilja säga att den är med i vårat team iallafall som ett litet behov iallafall. Och sen en till bara??

Jag: Jaa, det blir lite svårt för er nu....

R2: Jo men då är det väl ömdesidighet kring det arbete som utförs, det är ju också viktigt. Högt i tak, man ska kunna känna sig bekväm och sånna saker. De va dom tre, skulle jag välja just nu då.

Jag: Mm, ja men de låter bra. Då fortsätter vi tror jag. Vad anser du är viktigt när det kommer till team och programmering i kombination med varandra? Alltså hur man ska jobba som team och hur man ska programmera..

R2: Mm, mer hur man jobbar inom teamet då eller?

Jag: Jaa.

R2: Men det är väl för att om man säger att det finns olika nivåer av kunskap så är det ju väldigt viktigt att försöka sprida ut det man kan och då är ju parprogrammering, är ju en, är ju väldigt bra. Samma med mobprogrammering kan ju behövas ibland om man vill att det är någonting som verkligen ska gås igenom av allihopa. Men sen så finns det ju olika nivåer på det också om man har enklare arbeten, då ska man ju kunna jobba enskilt också men man måste kunna kombinera alla de här tre olika delarna och liksom känna av i teamet att ja men nu behöver vi nog köra lite parprogrammering för att sprida den här kunskapen och sen mobprogrammering eller är det liksom det här som ligger på den här frågan?

Jag: Jaa, men jag tycker det var ett bra svar. Tycker du att det finns någon problematik att som programmerare koda i ett team och varför isåfall?Några svårigheter eller så?

R2: Neej, de är väl, de kan ju bero mer på vem man är som person för att de finns ju faktiskt folk som vill kunna sitta helt enskilt och liksom koppla bort sig och tänka och sånt och så är det ju och då kan det ju vara svårt att jobba i ett team. För det är ju ändå ett visst, man vill ha liksom lugn och ro runt sig, för det är ju lätt att om man sitter i ett team så är det lätt att det blir lite prat och ljud och sånt. Men det är väl sånt som skulle kunna, som programmerare sitta i ett team. Ehm, sen finns det väl, har man inte, tar hand om teammedlemmarna, om det kommer någon ny som kanske inte riktigt känner sig säker på kodning kanske och om man inte liksom hjälper till med parprogrammering och sånna saker, då är det ju också, det hör ju till teamet att få till så att de får hänga med på utvecklingsfaserna där. Nu vet jag inte hur det är, i vårat team så har det inte varit något problem men det kan ju säkert vara så att i vissa gånger att man missar någonting i teamet, att ngåon medlem känner att de kanske halkar efter eller nånting. Så det kan ju vara ett problem.

Jag: Vi går vidare in på nästa del. Där vi kommer gå in mer på clean coding och det gemensamma tankesättet. Har du tidigare hört talas om begreppet clean code och vet vad det omfattar?

R2: Mm..

Jag: Jag har med den här bilden. *Visar modellen över de riktlinjer som ingår* Med de olika riktlinjer som Uncle Bob förespråkar och du hade lite koll på dessa?

R2: Jaa.

Jag: Har du tidigare tänkt på hur du skriver din kod? Om du kanske tänkt på hur du ger namn och så innan ni kom på att..

R2: Jaa, om man säger från skoltiden så tänkte man kanske inte så mycket på det, den första tiden där då när man höll på att lära sig programmering. Men sen har det liksom kommit med i bakhuvudet, man har ju sett sånna här kodexempel som man inte själv förstår, som vadå, dem här, hela metodnamnet kanske heter tre bokstäver och så finns det flera metoder som gör saker och variablerna in är helt obegripliga. Det är ett n eller ett i eller något sånt där. Så då, och det var ganska tidigt när jag började lära mig programmering och jag tyckte, jag gillade inte det konceptet riktigt. Det gick inte att läsa.

Och själv har jag inte, innan jag ens visste vad clean coding var så har jag försökt att men liksom för sin egen skull skriva så att kanske inte 100%, det var ju inte clean coding men det var mer så att man själv förstod metodnamnen och försökte vara tydlig med dem och sen, va heter det, vad var frågan?

Jag: Har du tidigare tänkt på hur du skriver din kod?

R2: Ja, så var det. Ja, precis. Så joo, men det har jag. Och sen har man då hållt på med labbar i senare år innan man började jobba också så då har det blivit lite mera, enhetstest har ju inte varit med, för det visste jag inte ens om att det fanns förräns nu när man gick i högskolan var det inte så mycket prat om det, men sen när man kommer ut och börjar se i arbetslivet, då kom ju det. Men just det med betydelsefulla namn var ju iallafall en tidig grej.

Jag: Tycker du att det är viktigt att skriva clean code? Att försöka skriva bra kod?

R2: Mm, jaa det är ju, jag tycker det nu ju mer när man har läst om det också och förstått fördelarna med att liksom det är ju inte bara det här med betydelsefulla namn utan det är hela konceptet runt det här med att få ner det till att en metod ska utföra en sak, alla de här sakerna runt omkring också att det gör de, man gör det för sin egen skull men även någon som kommer in och ska läsa det sen. När man läst det här från Uncle Bob att man ska kunna gå in i en klass och så läsa den här, det ska vara som att läsa en bok nästan. Att man ska kunna gå ner och varje metodnamn ska beskriva från första kapitlet till sista kapitlet, så det, jo jag tycker att det är viktigt, det ökar läsbarheten väldigt mycket och vad heter det det kommer in på nästa där varför det är viktigt att skriva clean code.

Jag: Jaa precis.

R2: Det underlättar ju framtida förändringar. Man vågar förändra, man kan se vad det är och förstå mer vad det är man kan förändra och kommer man då in med enhetstest på det också så ser man ju resultat och förändring väldigt tidigt.

Jag: Har nu någonsin tidigare.. Nu hade du kanske inte jobbat jättelänge med programmering men om du någonsin tidigare jobbat enligt ett uttalat sätt att skriva din kod på? Eller har det då kanske istället varit så att man sköter sin kodning och skriver så som man anser den vara bäst?

R2: Mm, jo men så har det varit när man lärt sig under skoltiden och sånt att det har ju inte varit på ett uttalat sätt utan man har gjort som det känts bekvämt på.

Jag: Tror du att ett uttalat gemensamt tankesätt, som då nu grundar sig i de här riktlinjerna inom erat team kan vara en fördel och varför tror du det?

R2: Ja, det gör jag! För att får man en gemensam grundförståelse inom teamet då, man kanske inte skriver på exakt samma sätt men man jobbar åt liknande håll vilket inom förvaltning kan vara, säg att man har, det kallas ju legacy code, tex när man har en äldre kodbas som ska göras om och då om man tillsammans har det clean coding-tänket då kan man gå in och se att ja, men det här måste vi göra om och då vet alla det i teamet att då går man in och ska göra förändringar så gör man det även på ett liknande sätt och man använder refaktorisering och vet vad det innebär. Så då, mm det är ju en fördel.

Jag: Tror du att det kan vara en nackdel? Att man har ett uttalat sätt att koda enligt?

R2: Njae, det beror väl på hur hårt uttalat man har. Alltså går man ner på, sätter man specifikt att, det finns ju i C# tillexempel och javan de har ju sina egna uttalade, så här skriver man metodnamn med stora bokstäver, stor bokstav första och liknande, klasserna ska ha ett namn och med properties skriver man på ett sätt och sånt. Förutom de så kanske man sätter några egna specifika saker, men är de för hårt styrda då kan det ju vara, då kan det ju vara, liksom låsa, då blir det ju egentligen inte clean coding heller utan då går man ju mer efter en mall igen då. Det är väl så, innan, det har ju varit så innan clean coding som tankesätt så har ju andra på något sätt försökt att liksom ha men att det ska vara såhär, det ska vara liksom tid där det kanske inte resharper fanns till hjälp, när man sa att man

måste ha en rads mellanrum mellan metoderna och så på sånna detaljnivåer. Då blir det för mycket också för en utvecklare. Ska man följa en sån checklista så då blir det ju inte heller nå bra.

Jag: Nej... nej. Hur tror du då att det här gemensamma tankesättet, som nu grundas i clean codes riktlinjer, kan hjälpa teamet att nå målet att skapa en lättläst och förvaltningsbar kodbas i kommande utveckling när ni ska göra om TRAP, göra det till ett ihopsatt system?

R2: Mm..

Jag: Hur tror du då att det kan hjälpa till att nå målet?

R2: Jaa.. Det var en svår fråga. Det beror på hur svårt svar man vill ha.

Jag: Nej, men svara det du tror så blir det nog bra, iallafall brukar det bli bra.

R2: Nej men om man säger såhär, har man inte jobbat så mycket med det så kan det ju liksom vara bra att printa ner ett gemesamt tankesätt först så alla förstår och sen när man väl kommer till att göra, nej men det blev svårt nu när jag läste skapa en lättläst och förvaltningsbar kodbas....

Jag: Jo men att om man jämför med hur det kanske varit i de gamla systemen och att målen då att få det nya systemet med lättläst och förvaltningsbart. Eller som ni beskrivit det iallafall och ja... att alla kanske följer dessa riktlinjer och att det ska skapa bättre kod typ.

R2: Joo, nej men absolut. Jag tror att bara man sätter och bestämmer vissa saker så och säger att, att följa grundläggande som single responsible principle och sånna där saker det är ju, det skapar ju och det här med enhetstest, tillsammans med sånna saker, och underhålla det också. Det hjälper både oss och vilka team det nu är i framtiden som ska sitta med det här. Så det finns ju ingenting som liksom säger emot att göra så nu. Om man säger att, man har ju själv sett sånt som från skolan och man har sett kod som inom jobbet och så med de här gigantiska metoderna tex, sånt kan man ju minimera om man bestämmer sig för att jobba med sånna här arbetssätt. Och sen är de väl att man försöker följa den där, va heter det, boy scout rule, från Uncle Bob. Att ser man någonting så ska man lämna den klassen lite snyggare, man behöver inte bygga om allt men gör någon lite grej, lite bättre.

Jag: Jaa. Då går vi vidare till nästa fråga. Vad kan du göra som individuell programmerare i ett team för att sprida tankesättet eller sprida kunskap om clean coding?

R2: Delvis så kanske om man läser någonting, hitta några böcker som informerar de andra att det här är bra information. Eller om man hittar någon annan info om det på nätet så är det ju alltid trevligt att kanske titta på det tillsammans och föra en dialog om att här finns det något bra tips. Sen är det ju som sagt att kör man parprogrammering så kanske man tillsammans bollar fram olika tankesätt både att sprida vidare. Eller om det är någon som är jättebra på det då kanske det är en sån som ska få sitta med parprogrammering tillsammans med de andra övriga teammedlemmarna för att sprida sin kunskap. Men som individuell programmerare så är det att jobba tillsammans med de andra. Det här med code review för att få andra att titta på koden är ju kanske inte, är ju kanske mer för att se om man missat någonting, inte riktigt för att sprida kunskapen genom code reviews, den tycker inte jag är så viktig. Utan det är ju mer bara för att dubbelkolla så att man inte låter någon bugg slippa igenom.

Jag: Vi fortsätter tror jag. Och då kommer det lite frågor kring hur ni jobbar idag med de här riktlinjerna och införandet. Har du idag börjat tillämpa några av de riktlinjer som clean code förespråkar. Om du försöker att tänka på dem?

R2: Jaa, precis. När man sitter nu och kodar så försöker man delvis då att refaktorisera om det man ser kan vara, så är det svårt att läsa det då, då kan det ju vara lämpligt att refaktorisera och bryta ut det till mindre metoder och förtydliga namn och variabler och liknande saker. Dom ställen där det är möjligt kanske man slänger på några tester, med enhetstest för att då kan man fortsätta att få hjälp av det sen

när man gör vidare förändringar. Så det är väl egentligen där vi, jag är just nu, när man gör någonting...

Jag: Hur anser du att det går? Ser du något problem i att tillämpa dessa riktlinjer? Finns det någon riktlinje som du tycker är svårare än nån annan att tillämpa?

R2: Njaa, det är väl egentligen, ibland så fastnar man väl på att tänka metodnamn. Det kan vara ganska svårt att ibland att tänka, vad gör den här och vad är det man ska få ut av, man kanske tror att det här namnet blir bra men sen är det inte ens i närheten av det korrekta namnet som den kanske borde ha. Så att det kan ju faktiskt vara lite svårt att få in det tankesättet tycker jag, iallafall just nu. Sen ibland kan det vara om man inte är så van vid enhetstest och det är ju också en sak att vänja sig vid att liksom hitta rätt ställen att sätta enhetstester på och sånt. I och med att vi inte jobbar med TDD tex så är ju det, det har man ju inte fått in som tankesätt än och det är ju ytterligare en nivå att börja jobba så. Men just att få sånna där saker runt omkring det här med att refaktorisera ut, det är inte svårt det tar lite tid bara. Och det ska det ju göra, man ska ju vara lite grundlig också. Men att komma på bra namn, det kan ju vara knepigare än man tror.

Jag: Jaa.. Finns det något arbetssätt som ni arbetar med i dag för att sprida kunskapen kring clean coding? Med arbetssätt så menar jag typ parprogrammering etc.

R2: Ja joo, men vi kör parprogrammering. Och sen har vi ibland också någon mobprogrammering och sen om det är någon som har mer koll på clean code så försöker vi ha någon dragning, berätta lite mer runt det. Hur den personen tänker så det är liksom sånna kontinuerligt sätt att sprida det. Och sen läsa tips och böcker. Läsa böcker...

Jag: Utöver de här då, arbetssätten som ni använder er av idag, tror du att det finns något annat arbetssätt som skulle kunna gynna, sprida kunskap eller att man lär sig mer? Eller känns det som att det ni använder er av idag är det de som gäller?

R2: Just nu vet man ju kanske inte, jag vet inte om det finns så många fler just nu då. Men samtidigt vet man ju inte. Mobprogrammering var ju nånting som några kom på bara på, jag tror det var i Silicon Valley ett team som testade att mobprogrammera och kom på att det här var ju en bra grej och sedan fick det en spridning men det är ju liksom, nej just nu så tror jag inte att jag har några andra förslag.

Jag: Nu ska vi se här... Varför tror du att de här arbetssätten kan vara en fördel för införandet av det här gemensamma tankesättet i teamet? Fördelar med dem här arbetssätten för att sprida tankesättet inom teamet? Det kanske vi redan pratat lite om redan..

R2: Joo, precis. Men att som att med parprogrammeringen att då är det ju två stycken som kanske till och med omedvetet börjar tänka på samma sätt. Mobprogrammering är väl mer för att få ut en lite mer bredare kunskap och sånna saker.

Intervju med respondent 3
2014-04-12

Jag: Hur länge har du jobbat med programmering?

R3: Mm.. ja heltid är det ju 1,5 år ungefär. Och så har jag ju jobbat lite från och till. Nån månad per gång, lite extra jobb.

Jag: Jaa, okej. Hur länge har du jobbat i teamet på Nethouse?

R3: TRAP-teamet har jag jobbat i sen jag började här vilket är 1,5 år.

Jag: Okej. Vilka är dina uppfattningar av att jobba i team? Då både positiva och negativa.

R3: Jaa.. Det är ju bra att vi är flera, så man behöver inte hålla koll på alla grejjer själv. Utan det är vissa som har lite olika specialiter. Man kan bolla ideér med varandra och det blir inte samma press på en person. Sen negativt de kan ju vara att det kan ju ta lite tid för att sätta arbetssätt och diskutera och få igenom det här så att alla är nöjda och överens och men det är ju sånt som ska ta tid så. Utvecklas och blir så bra som det går för teamet helst. Men det tar ju tid från allt annat.

Jag: Jaa.. då går jag vidare. Och det var det här jag tänkte kunde vara bra att du hade de här frågorna framför dig, för det finns åtta uttalade faktorer som anses vara viktiga för att ett team ska fungera bra och de här åtta faktorerna är *Läser upp de åtta faktorerna*. Då är min fråga till dig, vilka tre av de här faktorerna anser du är viktigast för ditt team och varför?

R3: Mmm.. Det är ju några, vilka tre som är viktigast... Det är ju lite..

Jag: Jaa, det finns ju.. alltså alla är ju viktiga på sitt vis men jag tänkte att det kan vara intressant om man behöver plocka ut tre stycken som man prioriterar högst då.

R3: Jag funderar lite men jag vet att anpassningsförmåga känner ju jag, den är ju viktig både med tanke på arbetssätt så man liksom inte spikar något som inte funkar alls utan att man kan arbeta om det och att alla är med på det och inte sätter sig emot det. Även just med, som nya tankesätt och så, så är det väl rätt viktigt när det kommer till just utveckling att det finns ju nya tekniker och det finns olika sånna bitar som kan vara effektivt att ta till sig för att göra arbetet bättre. Vad har vi mer... Ömsesidighet kring det arbete som utförs, det ser jag ju att den sista meningen, biten där, att lägga sig i andra teammedlemmars kod och även tvärt om. Det känns ju som en jätteviktig bit när man jobbar i ett team att det inte liksom, att det är ett gemensamt ägande av koden. Att det inte är en person som, ja men jag jobbar med den här lilla modulen eller vad det kan vara. Ingen annan får röra den. Jag gör på mitt sätt och det är jag som ändrar den. Det är ju helt fel tycker jag. Sen är det ju också viktigt att man får ju göra fel, man får testa sig fram. Det är ju inte alltid rätt. Man ska kunna våga prova saker också. Jaa.. vad har vi mer.. Inriktning kanske kan vara en viktig bit, men det är väl egentligen hela existensen för teamet, varför man är där. Men ja, det kan väl vara bra så man vet hur de ser ut, varför man är där och vad jobbar vi mot, varför gör vi det vi gör.

Jag: Jaa, men precis. De tre du tyckte var alltså ömsesidighet kring de arbete som utförs, inriktning och anpassningsförmåga var viktigt.

R3: Jaa.

Jag: Mm.. Jag tänkte bara så att jag upprepar dem så att jag förstod det rätt. Då fortsätter jag tror jag. Och då hoppar vi till nästa fråga som är vad anser du är viktigt när det kommer till team och programmering i kombination med varandra?

R3: Jaa..

Jag: Vi har ju pratat lite om att kanske lägga sig i andras kod och så vidare.. men vad mer kan vara viktigt när det kommer till team och programmering?

R3: Bra fråga... Det är väl en hel del men det är ju viktigt att man vågar fråga när man inte förstår kod, att man kan be om hjälp, att det finns folk som ställer upp och hjälper. Att det finns folk som ställer upp och diskuterar och bollar ideér och att man kan arbeta ihop.. Parprogrammering. Man sprider kunskap inom teamet, det är viktigt. Och försöka att få något gemensamt synsätt på vad, liksom hur är det vi jobbar, vad är det vi gör. Sitter alla utvecklare och skriver sin kod på sitt sätt utan att ha pratat med varandra så då kan det ju bli väldigt rörigt sen. Att man har liksom samma tankesätt och hur kod ska skrivas.

Jag: Mm.. Jaa. Tycker du att det finns nån problematik eller tycker du att det finns några svårigheter att som programmerare koda i ett team?

R3: Jag har inte så mycket annat att jämföra med.

Jag: Neej.. jag förstår.

R3: Ett projekt som jag jobbat med extra, extrajobbet innan jag började på heltid då. Då satt jag ju själv. Då tycker jag ju att det är bättre att jobba i team när man kan bolla ideér och inte liksom fastna på en grej som kanske inte fungerar så bra. Men jaa.. det finns det säkert men jag kommer faktiskt inte på någoting riktigt.

Jag: Du tror inte kanske att om det finns, om det är olka kunskapsnivåer på de som programmerar, att man har olika, man är olika duktiga, att det är något som kan vara ett problem?

R3: Neej. Inte egentligen att det är ett problem. Då får man ju försöka sprida kunskap till de andra ifall man kan mer eller om man kan mindre försöka suga åt sig av den kunskapen som de andra har. Det blir ju möjligheter istället.

Jag: Jaa, precis. Då tror jag att vi fortsätter till nästa del av frågor.

R3: Yes.

Jag: Har du tidigare hört talas om begreppet clean code och vet det omfattar? Vilka riktlinjer som ingår?

R3: Ja. På universitetet har jag läst böcker och intern kurs också.

Jag: Mm. Men du har iallafa lite koll på det?

R3: Mm.

Jag: Har du tidigare tänk på hur du skriver din kod? Tänker du på hur du skriver din kod när du programmerar?

R3: Jaa, efter man hörde talas om clean code så absolut. Innan, inte tillräckligt. Men då satt jag ju mest själv. Som jag ser det clean code är ju viktigare när det ska förvaltas, när andra ska kunna förstå koden enkelt. Och sitter du och latjar hemma eller någonting så är det ju inte lika viktigt. Då kanske ingen någonsin kommer se det, så då spelar det ingen roll hur den ser ut kanske inte är lika stort heller.

Jag: Då kanske nästa fråga, eller de två nästa frågorna går in på lite det du pratat om. Varför det är viktigt att skriva clean code och varför du tycker det är viktigt att skriva clean code. Du pratade ju lite om förvaltning där.

R3: Precis, det är väl en av de viktigaste bitarna med att skriva clean code är ju just att, jobbar man i ett team eller jobbar man med en kodbas så är det ju inte alls säkert att det är jag som kommer använda eller fortsätta utvecklingen eller liksom in och pilla där det är jag som skrivit min kod utan det är ju

viktigt att det ska vara lätt att förstå för i stort sett vilken utvecklare som helst. Så att de inte blir så inlåst att nej, men det där är hon eller hans lilla kodsnutt. Det är han/hon som får ta det. Det går ju mot vad jag tycker. Det ska vara gemensamt ägande på kodbasen så att vem som helst kan ändra vart som helst. På alla nivåer, även att alla ska kunna förstå den. Och sen så är det ju viktigt också med, när man designar sin kod så att den inte spretar så mycket också. Att det blir single responsibility. Att vi inte har några få stora klasser som gör allt och att det blir jättesvårt att hitta. Utan att man delar upp det där och då blir det enklare för om man ska in och ändra eller titta på vad som händer.

Jag: Jaa, har du någonsin tidigare jobbat enligt ett uttalat sätt att skriva din kod på? Eller har det då varit mer att var och en sköter sin kod, att man skriver på det sätt som man tycker lämpar sig bäst?

R3: Jaa, nej.. Jag har väl egentligen inte någon riktigt uttalat sätt nu heller. Men det är ju lite svårt också just med TRAP, för att den kodbasen är ju jätterörig och man vill ju inte ändra alldeles för mycket för att då blir det så mycket mer och testa på den existerande kodbasen. Men när man skriver nytt så man väl tänkt liksom göra det förvaltningsbart. Det är ändå teamet som ska kunna ändra på grejerna. Men jag har väl inte riktigt, inte här i Örebro att vi har pratat ihop oss, jag vet inte hur de är i Borlänge om de är bestämt något hur vi ska göra.

Jag: Att det inte är det?

R3: Nej, jag tror inte vi har något uttalat sätt. Inte som jag har varit med och hört.

Jag: Neej.. Men inte då att ni ska tänka mer och skriva enligt clean coding eller så? Jag tänker att jag ser det som kanske att man pratar om det och att det blir mer uttalat så.

R3: Joo, absolut. Det tror jag väl vi gör. Utan att prata så mycket om det ändå. Det är väl vad som händer om vi inte gör det. I resten av kodbasen. Det tar lång tid att hitta de små snuttarna man är ute efter, istället för att ja, men det är ju den filen och den här metoden tar kanske en minut att hitta. Medans det i den existerande kodbasen kan ta en halvtimme, timme att hitta.

Jag: Jaa.. Jag kommer gå vidare nu. Tror du att om man har ett uttalat gemensamt tankesätt som nu grundar sig i de här riktlinjerna, kan vara en fördel när man jobbar i ett team såhär med programmering?

R3: Jaa.. Det tror jag väl. Det är ju viktigt att alla gör på samma sätt också. Alltså ifall en inte är med. Om vi har ett gemensamt tankesätt från början och sen så gör alla det här ändå förutom ett par, säger vi. Då blir det ju likadant tillslut ändå. Då blir det ju vissa bitar som inte är förvaltningsbara. Det är inte alla som kan gå in och ändra och då blir det tekniskt strul och det blir alla som inte kan ändra koden kanske, det tar för lång tid att förstå.

Jag: Ser du några nackdelar då med att ha ett uttalat sätt som man kodar efter eller försöker iallafall tänka på?

R3: Jaa, det beror ju lite på.. Jag tänker ifall, så länge man inte, att det blir som några regler att såhär måste man göra så att det blir ett hinder. Man liksom kan krångla till koden i onödan med att dela upp den tillexempel i massa metoder fast det då egentligen inte kanske behövs. Det är ju precis som det står här, riktlinjer. Så länge man har det som riktlinjer och inte som regler så tror jag inte att det är några större nackdelar.

Jag: Hur tror du att det här gemensamma tankesättet kan hjälpa teamet att nå målet att skapa en lättläst och förvaltningsbar kodbas? För det är väl lite målet med egentligen om man jämför med hur det varit tidigare att det kanske vart svårt att förvalta för att det varit långa metoder osv. Att man då med ett gemensamt tankesätt vill uppnå målen, hur tror du att det kan hjälpa?

R3: Nja men jag tror väl att det viktigaste är väl att alla har med sig det att vi ska inte ha en metod på två hundra rader kod. Det går ju inte att hantera på ett enkelt sätt. Sen också viktiga bitar som att inte

ha någon kodduplicering här och var. Det är ju också en väldigt viktig bit som inte existerar i dagens kodbas. Där finns det ju, vi ska ändra en liten funktion men då är den utspridd eller så finns det samma logik på tre olika ställen runt om i hela kodbasen. Då måste man ändra på alla ställen. Tänker man på de här bitarna när man skriver sin nya kod och när man ser att här ligger det samma grejjer på tre ställen, att man kanske bryter ut de och ser till att det kanske samlas på samma ställe så blir det ju mycket enklare, det blir mindre risk att man råkar missa någonting när man väl är där och ändrar nästa gång. Det blir inte lika utspritt.

Jag: Vad kan du göra som individuell programmerare för att sprida kunskap och för att sprida tankesättet inom teamet?

R3: Jaa. Det viktigaste verktyget för att sprida kunskap bra, som jag ser det just nu är ju att parprogrammera. Man sitter ihop och man kan diskutera varför gör man sådär, här kan man ju faktiskt bryta ut en metod så blir det ju lite enklare att läsa. Och jag tror att.. Eller jag känner iallafall att parprogrammering ger rätt mycket mer än att typ ha någon dragning, eller vad man nu annars kan tänka sig. Diskussioner kan man ju ha, hela teamet. Men sen behöver man ju väl också sätta sig ner och göra det också. Just parprogrammering är ju bra för då kan man ju ha en diskussion samtidigt, som jaha nej men det där har inte jag tänkt på, och så kan man ju göra.

Jag: Jaa.. Vi går vidare tror jag! Och då är det lite, den sista delen, och det är väl lite om hur ni gör idag. Hur långt ni kommit och så. Har du idag börjat tillämpa några av de riktlinjer som clean code förespråkar och vilka isåfall? Tänker du mer på riktlinjerna och har dem i bakhuvudet och försöker tänka på det när du programmerar?

R3: Joo, absolut. Det är ju mest när vi skriver ny funktionalitet vi har tid med det. Dock i den existerande kodbasen så kan man ju alltid refaktorisera och byta namn på variabler så att de heter något vettigt så att man förstår vad det är istället för att behöva läsa vart de deklareras, för att förstå vad det är egentligen. Nya grejjer så brukar jag iallafall tänka på att inte göra metoder för stora, det ska vara vettiga namn och SRP. Att man inte samlar massa olika funktionalitet i samma klass utan endast utföra en grej. Då vet man vart man hittar det sen. Och sen som refaktorisering som att man inte har kodduplicering, man samlar det på ett ställe där de andra även kan anropa från istället för att dela upp. Se till att det inte finns några kommentarer, det brukar kunna vara rätt bra.

Jag: Tycker du att det går bra att tänka enligt riktlinjerna eller tycker du att det känns svårt och finns det några riktlinjer som du tycker är svårare att tillämpa än andra?

R3: Det går väl bra. Det är väl mest så att det är ju en period i början då man lärt sig och vill göra det på allt, att man nästan ser det som regler och det är väl det som är, kan vara ett problem isåfall. Om man går ända ner till att lyssna på Uncle Bob som säger att en metod bara ska vara fyra rader kod och se det som en regel, då kan man ju få problem.

Jag: Jaa, jag förstår det.

R3: Men i den existerande kodbasen, man man tänker D i SOLID. Dependenci inversion. Det är ju ingenting som är byggt så. Utan allting är ju beroende av konkreta klasser istället för att vara beroende av interface tillexempel. Och till det tänket så kan man ju bygga om vissa bitar av systemet från grunden egentligen. Men annars så själva grunden, när man skriver själva koden att den blir lättläst och se till att man själv förstår den, om det kan vara något som någon annan kan missförstå. Försök tänka om. Det ligger väl mest med nu bara.

Jag: Finns det något arbetssätt ni använder er av idag för att sprida kunskapen kring clean coding?

R3: Jaa, det blir väl det när man parprogrammerar tänker jag. Men annars ingenting, jag sitter ju rätt själv här i Örebro. De hade ju ett större team här i Örebro förut men just nu så är det ju bara parprogrammering.

Jag: Tror du att det finns något annat sätt som skulle kunna gynna införandet av det gemensamma tankesättet, ett annat sätt som kan hjälpa till att sprida kunskap, att hjälpa till att komma in i tänket?

R3: Ja, det kommer väl in lite på det jag sa förut va.

Jag: Parprogrammering?

R3: Jaa, men det går ju att ha dragningar också om det skulle behövas om vissa känner att dem inte har koll på alla bitar men jag tror ju fortfarande att parprogrammering ger mer för varje person. När man kan sitta och diskutera och fråga allting samtidigt som man gör grejjerna. Istället för att sitta och lyssna för i teorin så är ju allting jättelätt.

Jag: Ja men precis.. Varför tycker du att de här arbetssättet är en fördel? Nu kanske vi har pratat lite om det innan då, det med att man får sitta med varandra och diskutera och så. Varför tycker du det är en fördel med de arbetssättet?

R3: Ja, nej men det är väl mest att det blir både, eller kan bli beroende på hur mycket man diskuterar och pratar men att det blir teori och praktik samtidigt. Istället för att komma ihåg saker i efterhand när man väl sitter med något problem någonstans att det faktiskt ser ut såhär. Hur kan jag göra? Eller så lär jag mig bättre. När man får praktisera grejjerna. Teorin kan ju vara med men den brukar ju oftast vara liksom förenklad. Till exempel som rätt simpla grejjer som såhär kan man göra, gör inte såhär. Men så ser det ju kanske inte ut när man väl sitter och utvecklar på riktigt.

Intervju med respondent 4
2014-04-12

Jag: Hur länge har du jobbat med programmering?

R4: Jag har jobbat här i en vecka bara. Och jag jobbar väl egentligen som testare, eller mitt huvudsyfte kommer att vara testning.

Jag: Ja, okej.

R4: Men jag har ju även jobbat lite som databasadministratör i Stockholm där jag kodat i SQL och så om det räknas.

Jag: Jaa, men de säger vi att det gör. Hur länge har du jobbat i teamet på Nethouse?

R4: Ja, också en vecka.

Jag: Jag vänder på frågan lite eftersom du inte jobbat så länge i teamet. Men vilka är dina förväntade upplevelser av att jobba i ett team? Positiva och negativa. Nu kanske du har jobbat i team innan också?

R4: Njaa, jaa. Team fast vi var ändå ensamma om våra uppgifter, vi satt i ett lag med vi hade olika roller. En var .NET windows, en var oracle och en va linux. Eh jaa, förväntningar det är väl att vi är en grupp som arbetar, vi är inte själva i det vi gör och beslut vi tar. Vi har en gemensam leverans vi levererar någonting gemensamt, att det inte är att du har gjort det här och du är ansvarig för det utan får lösa allting tillsammans och kommunicera mycket med varandra. Det är väl min känsla av att det är så tycker jag, att det verkar som iallafall. Det jag varit med om är man pratar om att alla bidrar och det är mycket så här att de hjälper varandra tillsammans i uppgifter och så att de parprogrammerar eller löser uppgifter i grupp och sånt.

Jag: Det är lite det positiva då?

R4: Jaa..

Jag: Tror du det kan vara något negativt med att jobba i team?

R4: Jaa, inget jag funderat på men man kan ju tänka sig att det kan uppstå konflikter i grupp och att folk inte kommer överens eller att man har olika sätt man vill lösa något på och sånt. De är väl det isåfall som skulle vara det negativa. Att man hamnar i konflikter med någon.

Jag: Ja, nej det är väl aldrig roligt kan jag tänka mig.. Vi går vidare. Det finns åtta uttalade faktorer som anses vara viktiga för att ett team ska fungera bra. *Läser upp de åtta faktorerna* Alla faktorerna är ju viktiga i sig, men för att göra det lite svårare så ska du få välja ut tre stycken som du tror är viktigast.

R4: Ledarskap skulle jag vilja säga är viktigt. Att det finns någon person som faktiskt styr skeppet när det blir tungt eller att någon har auktoriteten att bestämma och säga att såhär gör vi så att det kanske händer någonting. För den tycker jag går ihop lite med back-up, att ledarskap nästan täcker upp den delen. Någon som bestämmer. Men jag väljer inte back-up! Utan jag väljer ledarskap. Jaa, anpassningsförmåga skulle jag också välja. Det är väl viktigt att man kan anpassa sig annars hamnar man kanske lite i vattenfallsmodell, liksom att allt blir lite för uppställt och styrt. Och sen ömsesidigt förtroende, att alla tar sin roll i teamet liksom att nån inte gör någonting eller att någon blir belastad med mycket och att man slutför sina uppgifter och kan man inte så kan man ta hjälp av någon annan. Förtroende för varandra är väl bra, att man kan lita på att saker blir gjorda. De tre känns som de viktigaste som jag skulle välja.

Jag: Jaa, då går vi vidare då. Vad tror du är viktigt när det kommer till team och programmering i kombination med varandra?

R4: Hur menar du då?

Jag: Jaa.. Hur menar jag då.. Men om man tänker då att man ska programmera ihop också vad är viktigt då? Kan det vara kanske kommunikation eller liksom något annat viktigt som du tycker man behöver tänka på? Eller som du tycker är viktigt?

R4: Det viktigaste är väl att koden är begriplig för alla liksom att man kan sätta sig in i den snabbt utan att den personen behöver vara där, att den inte gjort någon konstig fulkod som bara den personen förstår, någon sträng som inte betyder någonting, att här står det något obegrepligt som man inte kan förstå. Den är väl viktig. Att det finns nån sorts versionskontroll så att man kan göra back-up och ändra om någonting är fel eller sånt.

Jag: Mm.. Tror du att det finns något problem eller svårighet att som programmerare koda tillsammans i ett team? Att man sitter flera stycken och jobbar ihop istället för att sitta enskilt.

R4: Hmm.. Neej. Jag tror nästan bara att de är finns fördelar mest. Så är det ju inte men.. vad skulle det kunna vara.. Är det två styckna som har väldigt låg nivå så kanske det inte bidrar så mycket och då kanske de är bra och mixa upp det med någon som är lite mer erfaren som kan lära ut och att den då inte blir för dominant heller då och gör alltid. Nej, jag vet inte faktiskt...

Jag: Vi fortsätter då till nästa del.. Har du tidigare hört talas om begreppet clean code och vet vad det omfattar?

R4: Neej, eller jag har hört talas om begreppet ganska ofta men jag är inte helt hundra på vad det betyder. Men jag tror jag har lite koll..

Jag: Jag har en liten bild här med alla de riktlinjer som Uncle Bob förespråkar. Du kanske hört talas om Uncle Bob?

R4: Jaa. Det känner jag igen.

Jag: Jaa, det där är iallafall det som han förespråkar. Och meningen med den här bilden var att visa bara så att vi pratar om samma sak, att begreppet omfattar dessa riktlinjer. Men jag tänkte att vi går vidare.. Har du tidigare tänkt på hur du skriver din kod? Nu hade du kanske inte programmerat jättelänge men...

R4: Nej, jag har inte kodat jättelänge så. I viss mån har man väl gjort det. Men främst har det ju alltid varit att få det att fungera, så man har inte tänkt på att man ska lämna över det till någon annan för att jag har ju inte programmerat i grupper där man lämnar över sitt arbete så. Mest vart kodning för mig själv. Så att det har jag inte tänkt på så jättemycket. Få det att fungera bara.

Jag: Jaa joo, jag kan känna igen det där när man bara vill få det att fungera.. Tycker du att det är viktigt att skriva clean code? Eller tror du att det är viktigt att göra det?

R4: Ja men det tror jag är väldigt viktigt för levnaden av koden. Visst det går att programmera dirty code, att du får det att fungera men sen kommer det ju alltid behöva uppdateras och förbättras, förnyas och gör du det snyggt och cleant från början så kommer det nog bidra till förvaltning går bättre.

Jag: Har du någonsin tidigare jobbat enligt ett uttalat sätt att skriva din kod på? Du får svara utifrån det du kan. Tycker du kanske att det vart så att var och en sköter sin kod på det vis som de anser den vara bäst på?

R4: Ja, neej. Jag har ju jobbat i skolan då när jag pluggade. Jobbat i grupp och försökt kodat, men då var det ju inte, vi skulle komma fram till en lösning och alla skulle koda vissa delar och vi skulle göra uppgiften tillsammans med de blev oftast att det blev den enas uppgift som man använde hela tiden för att koden som de övriga skrev inte synkade med varandra, de såg lite olika ut och sådär.. Vad var frågan?

Jag: Om du tidigare jobbat enligt ett uttalat sätt att skriva din kod på?

R4: Nej, vi hade inget uttalat sätt hur vi skulle göra liksom.

Jag: Nej.. Men tror du att ett sånt här gemensamt uttalat tankesätt inom ett team kan vara en fördel?

R4: Ja, det tror jag absolut. Sen om man ska följa det slaviskt så är det ju en annan sak. Men att ändå ha riktlinjer brukar ju alltid vara bra att ha. Så att det tror jag. Men det gäller också att alla är införstådda på vilka riktlinjerna är också. Så att man inte har olika synsätt på riktlinjerna känns de som.

Jag: Tror du att det kan vara en nackdel?

R4: Inte vad jag vet. Ehm, det jag skulle kunna tänka mig är, ja det kan väl hända att man låser in sig på något men jag kan inte komma på vad det skulle kunna vara.

Jag: Okej, men tror du då att ett gemensamt tankesätt, som nu grundas i clean codes riktlinjer, kan hjälpa ert team att uppnå målet att skapa en lättläst och förvaltningsbar kodbas?

R4: Jaa, det tror jag ju. Absolut.

Jag: Vad kan du göra om individuell medlem i teamet för att sprida tankesättet eller sprida kunskapen om du kan någonting? Hur skulle du kunna göra då?

R4: Det är väl att försöka förmedla den kunskapen, som när vi parprogrammerar tillsammans eller kör en mobprogrammering där alla är delaktiga och komma med förslag. Och visa på att det skulle vara bättre att köra på det här sättet eller liknande.

Jag: Jaa.. Vi går vidare till nästa del tror jag. Den handlar lite mer om idag, och hur det ser ut idag. Har du idag börjat tänka eller tillämpa några av de riktlinjer som clean code förespråkar och vilka isåfall? Finns det några du tänker på när du väl sitter där och skriver din kod?

R4: Nu vet jag inte riktigt vad riktlinjerna är inom clean code, uttalat men alltså om jag tittar på den här modellen här så..

Jag: Betydelsefulla namn kanske, tänker du lite på sånna saker eller tänker du något på metoder eller så?

R4: Jo, jag tänker ju på att försöka vettiga namn på alla metoder och klasser och så som stämmer överens med vad den ska göra. Kommenterar utifall det skulle behövas. Jaa, det är väl det isåfall.

Jag: Ser du något problem med att tillämpa riktlinjerna eller tycker du att det finns någon svårighet med det? Finns det någon som är svårare än nån annan?

R4: Jaa, vissa kan ju vara svåra pga min brist på erfarenhet, alltså hur man formaterar eller bygger upp rätt struktur och sånt där. Och att för mig som inte har så mycket erfarenhet kanske det tar mycket längre tid att göra det än att göra en snabbkodning som jag kanske klarar av att göra. Man har ju oftast leveranstider på saker, saker som ska vara klara vid en viss tidpunkt så då kan det ju förhindra.

Jag: Mm, jaa. Finns det något arbetssätt som ni idag använder i teamet för att sprida kunskapen kring de här ritklinjerna?

R4: Inte vad jag vet..

Jag: Inte? Nej.. Då kanske det blir lite svårt att svara på de kommande frågorna. Du pratade ju om parprogrammering...

R4: Jaa, jag tänkte precis det nu. Nu förstår jag vad du menar med arbetssätt. Ja men då har vi ju parprogrammering, mobprogrammering. De är väl ett par arbetssätt som vi använder oss av. De har

berättat att det har mycket interna möten och så där de som vart på utbildning får prata om vad de lärt sig och gjort och sprider sin kunskap. Hur vi kan anpassa oss och så.. så joo, det finns ju.

Jag: Tror du att det finns några ytterligare arbetssätt då som man skulle kunna använda och som skulle kunna vara en fördel? Något du tänkt på?

R4: Njaa, det är inget jag funderat på kanske. Föreläsningar, genomgångar av det man lärt sig och speciellt när man tar in nya att höra vad de tycker och om de har några nya ideér.

Jag: Varför tror du att tex parprogrammering och mobprogrammering kan vara en fördel när man inför ett nytt gemensamt tankesätt, som i de här fallet börja koda enligt vissa riktlinjerna, varför tror du det är en fördel då?

R4: För då får man ju ta del av en annans åsyn av dem. Så man inte uppfattar saker olika. Riktlinjen är det här och då kanske det inte är så att det är såhär de menar med den. Att man får samma syn på det. Eller att man kanske har missat någonting. Att den här riktlinjen innebär att vi måste skriva koden såhär och att någon annan då kanske inte tänkte på det. Man får mer input från någon annan. Sen kanske man kan byta ut den man parprogrammerar ihop med till någon annan för att se hur den gör osv. Då ser man ju hur andra tänker också vilket nog gör mycket.

Intervju med respondent 5
2014-04-24

Jag: Då tänkte jag börja med att fråga hur länge du har jobbat med programmering?

R5: Oj, jaa.. sedan 92, 93 någongång.

Jag: Jag har förstått det som att du har en lite mer ledande roll i teamet?

R5: Jaa, precis. Jag har fått arbeta som leadrollen.

Jag: Jaa, hur länge har du jobbat i teamet på Nethouse?

R5: Det är väl nu i 1,5 år. Det är väl sedan i januari 2016.

Jag: Vilka är dina uppfattningar av att jobba i team? Både positiva och negativa. Har du jobbat i team förut?

R5: Jag har många olika erfarenheter av att jobba i team men det här teamet tycker jag fungerar jättebra. Det finns ju mer eller mindre bra fungerande team.

Jag: Jaa, så är det väl. Det finns iallafall åtta uttalade faktorer som anses vara viktiga för att ett teamwork ska fungera bra. Om du tänker på dessa faktorerna i samband med Clean Coding och om man ska börja använda sig av dessa riktlinjer. *Läser upp de åtta faktorerna* Utav dessa åtta faktorer så ska du få välja ut tre av dessa som du anser vara viktiga i sammanhanget med Clean codes riktlinjer.

R5: Ömsesidigheten är ju bra.

Jag: Mm.. Är det ömsesidigheten kring det arbete som utförs?

R5: Ja, precis att man törs läggs sig i och att man inte liksom har den här, att det är min kod utan att det är våran kod som man hjälps åt att förvalta. Det är väl A och O inom clean coding. Att tänka att det är våran kod och inte min. Sen har du ju lika den här ömsesidigt förtroende, att lita på att du skriver kod som jag kan göra. Att vi sätter oss tillsammans, parprogrammerar eller så kan var och en gå och skriva varsin kod. Så kan man kasta kod till varandra sen. Och lika den tvåvägskommunikation, att man törs fråga om det skiter sig. Det är väl dom tre som jag känner är viktigast, iallafall för clean coding. Alla andra är ju viktiga på andra vis men..

Jag: Jaa, men precis. Det är ju ändå åtta faktorer som anses viktiga så det är ju klart de är viktiga allihopa. Tycker du att det finns någon problematik med att koda i team? Och varför isåfall?

R5: Problematiken som kan vara det är väl att alla tänker olika från början iallafall. Men då gäller det ju att jobba ihop sig som ett team. Det är ju ungefär som att släppa ut elva stycken fotbollsspelare som aldrig spelat ihop, det går ju inte heller så bra.

Jag: Ja, men precis. Det finns väl kanske riktlinjer man följer där också.

R5: Jaa, det finns ju det. Man har en tränare som talar om att nu går du dit, dit, dit. Det tar ju nästan en hel sommarsäsong innan de hittar varandra.

Jag: Jaa.. Vi går vidare till del två som handlade lite mer om clean coding och det gemensamma tankesättet. Hade du tidigare hört talas om begreppet clean coding och vet vad det omfattar?

R5: Jaa. Ungefär ja.

Jag: Okej, har du tidigare tänkt på hur du skriver din kod? Jag tänkte eftersom ni nu ändå bestämt er för att försöka skriva clean code men har du tidigare tänkt på hur du skriver din kod?

R5: Det har ju.. det har ju funnits så mycket olika regler genom tiderna. Från början så skulle det vara så få funktioner som möjligt. Iallafall om man kodade PHP, för det tog en massa minne. Nu har ju begreppet Clean Code kommit och att nu finns det mycket minne också så att nu ska man ha så många funktioner som möjligt så att alla ska förklara vad dem gör, att de gör så lite som möjligt. Man har ju inte alltid kunnat gjort clean code.

Jag: Som Uncle Bob förespråkar då?

R5: Mm.

Jag: Men du tycker att det är viktigt att skriva clean code?

R5: Jaa, alltså att komma in i ett projekt och sätta sig ner och läsa kod som är en enda stor röra va, det tar en himla tid att komma in i. Så att, om man skriver det som en bok, ja men då är det lätt att komma in i koden.

Jag: Du pratade lite tidigare om att du jobbat med PHP och att det då skulle vara så få funktioner som möjligt och hit och dit. Nästa fråga handlar om du någonsin tidigare jobbat enligt ett uttalat sätt att skriva din kod på? Eller har det vart mer att man skriver sin egna kod och man sköter sitt?

R5: Man har ju försökt att följa de riktlinjer som varit men det är ju trender det också. Som sagt i PHP så skulle det vara så stora funktioner som möjligt i princip och man ska försöka skriva någon form av objektorientering i något språk som inte ens är objektorienterat. När man kodar i C# ska det se ut så och kodar man i stora datormiljöer så då är det kolumnbaserat så då är det noga att du håller dig på rätt, det är noga vart du skriver koden också.

Jag: Jaa, det är mycket att tänka på! Men tror du att ett gemensamt tankesätt i ett team, som grundar sig i clean code riktlinjer kan vara en fördel?

R5: Jaa, oja! Code reviews och parprogrammering också.

Jag: Kan det vara en nackdel då?

R5: Nja, det tar ju lite längre tid att sätta standarderna innan alla kommer in och kommer det in någon ny, en ny spelare så måste ju den lära sig språket också. Men när man väl får till det så är det ju självsmörjande så att säga.

Jag: Mm, jaa det är väl det. Det tar ju tid innan man kommer in i det, men när man väl gör det så kommer det ju vara värt det. Hur tror du att det gemensamma tankesättet, clean codes riktlinjer, kan hjälpa teamet att nå målet att skapa en lättläst och förvaltningsbar kodbas?

R5: Ja, får man en funktion som talar om vad den gör istället för bara ett konstigt namn så då är det ju ganska självförklarande. För då slipper man ju skriva en masssa dokumentation.

Jag: Jaa för kommentarer vill man väl kanske inte ha heller?

R5: Det är ju det första som blir gammalt.

Jag: Jaa, precis det är svårt att hålla ordning på dom. Men vad kan du, som individuell programmerare i teamet, göra för att sprida det gemensamma tankesättet och kunskapen du har kring dessa riktlinjer?

R5: Jaa, man får ju köra parprogrammering eller så kör man en mobprogrammering om det är större grejjer. Eller code reviews för att sprida språket.

Jag: Jaa. Vi går in på del tre tror jag och det handlar lite mer om hur ni jobbar idag med att använda dessa riktlinjer. Och har du idag börjat tillämpa några av de riktlinjer som clean code förespråkar? Är det någon särskilt du tänker på eller som liksom har satt sig redan nu?

R5: Jaa, jag försöker ju hålla ner så att varje metod bara gör en sak. Ser jag någon som gör mer än en sak så bryter jag isär den. Det är ju boy scout rule, den kör vi. Och lika försöker vi ge dem väl beskrivna namn. Funktionsnamn och klassnamn.

Jag: Hur tycker du att det går? Är det något som är svårt eller är det något problem med att tillämpa riktlinjerna? Eller iallafall försöka, nu kanske man inte ska se det som regler men vissa kan ju vara svårare än andra ändå.

R5: Ibland är det ju, det är ju svårt att hålla reda på har jag skrivit det här förut? Finns den här metoden någonstans. Som när man bryter isär mycket, tillslut så har man ju skrivit samma kod på fyra olika funktioner och hittar det. Det är det som är det svåra. Förut så vad det ju inte så noga när allt låg i samma funktion. Men nu så bryter man isär det så vill man ju inte ha samma funktion med tre olika namn.

Jag: Nej, men precis. Men nu när ni ska börja arbeta enligt dessa riktlinjer, finns det något arbetssätt som ni använder er av idag?

R5: Jaa, vi kör ju då parprogrammering och och grupprogrammeringar. Mobprogrammering.

Jag: Finns det något annat sätt som du tror skulle kunna vara bra? Som du tänkt på?

R5: Jaa, vi skulle kunna använda code reviews mer. Så att innan man checkar in koden, att man skickar vidare för en kontroll då.

Jag: Att man får läsa igenom varandras kod då?

R5: Jaa, det gör vi inte så mycket som man skulle kunna göra.

Jag: Neej. Varför tycker du att code reviews skulle vara bra att använda sig mer av?

R5: Det är ju olika tillämpningssätt. Är det, ska man lösa ett problem så är ju parprogrammering bättre. För två hjärnor är ju bättre än en. Om var och en sitter på sin kammare och gör något enkelt så kan det ju vara bra att skicka koden för lite code review bara om det är enklare grejjer.