Designdokument
Bemanning i GOLI Kapacitetsplanering

Niclas Olofsson

12 mars 2014

1 Inledning

GOLI Effektivitetsplanering är ett system för produktionsplanering, vilket används för att planera och följa upp produktionen i en viss verksamhet. Ett exempel kan vara en röntgenavdelning på ett sjukhus, där man planerar att göra ett visst antal undersökningar per dag. Vissa dagar lägger man in extrapass med syftet att korta ner vårdköerna, och vissa dagar måste röntgenmaskinen genomgå underhåll. Detta leder till planerade variationer i antalet undersökningar över tid, vilka kan följas upp genom att jämföra det planerade antalet med hur många undersökningar som faktiskt gjordes. På grund av efterfrågan från kunder vill GOLI utöka sin tjänst till att även innefatta stöd för hantering av bemanning.

Syftet med detta dokument är att beskriva funktionaliteten i den tänkta utökningen av systemet enligt de önskemål som kommit från kunder, samt beskriva en plan för utveckling, tester och implementation av denna.

2 Bakgrund

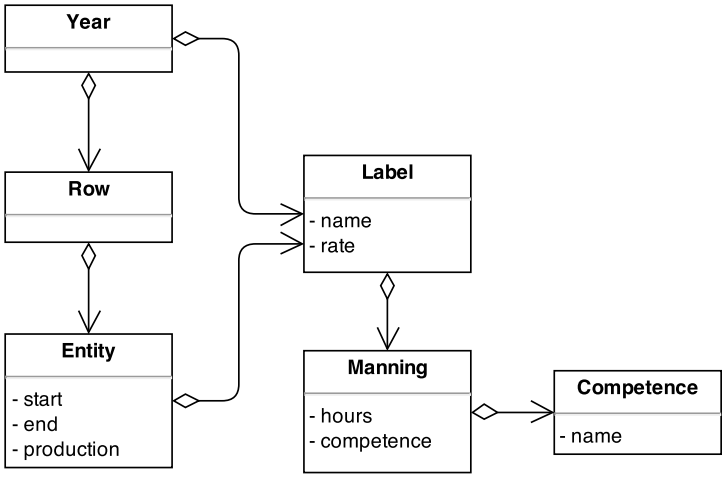
GOLI används i nuläget av verksamhetschefer, eller i vissa fall chefer för ett visst röntgen-laboratorium, för att göra och följa upp produktionsplaner. Totala produktionsbehovet räknas fram manuellt utifrån faktorer som t.ex. antal kvinnor i länet i fallet med mammografi, samt statistik från föregående år. Detta ligger sedan till grund för hur många timmar som man ska bemanna med per dag, vilket i sin tur används vid schemaläggningen - d.v.s. när de planerade timmarna ska fördelas på olika personer.

I planeringsprocessen är man intresserad av hur en förändring av de planerade aktiviteterna under året påverkar mängden timmar, så att man kan höja produktionen för året utan att behöva schemalägga fler timmar. Ett exempel kan vara att uppstartstiden för varje arbetspass är konstant, vilket gör att man kan uppnå högre produktion genom att ha några få långa pass istället för många korta pass. För att kunna experimentera med olika sådana planer krävs att man kan tilldela ett visst antal timmar till varje aktivitet, samt få dessa summerade.

Eftersom att det planerade antalet bemannade timmar per dag ligger till grund för schemaläggning, vore det även önskvärt om denna planering gav ett bra underlag vid schemaläggningen. Man skulle på sikt även vilja utöka stödet för bemanning för att kunna ge en mer detaljerad bild av vilken personal som behövs. Vissa undersökningar kräver olika typer av kompetenser, vilket innehas av olika typer av personal. Detta problem kan dock i realiteten vara mycket komplext och ryms inte inom ramen för examensarbetet, men bör beaktas vid design av den grundläggande funktionaliteten för bemanning.

3 Metodik

Då examensarbetets fokus är testning och testbarhet, kommer testdriven utveckling (TDD) att användas vid implementationen av den nya funktionaliteten, med vissa inslag från behaviour-driven development (BDD). Syftet med detta är att få en bild av hur väl detta fungerar i praktiken, samt vilka svårigheter och utmaningar som finns vid testning av webbaserade applikationer. Automatiska systemtester kommer att



Figur 1: UML-diagram för de relevanta delarna av systemet

göras med Selenium och Capybara, unit-tester med RSpec och Jasmine. Själva implementationen kommer att ske i Ruby och CoffeScript med ramverken Ruby on Rails 4 och KnockoutJS 3, med MongoDB som DBMS, då detta är de teknologier som används i det befintliga systemet.

4 Systemdesign

4.1 Datamodeller

Ett förenklat UML-diagram för de relevanta delarna av systemet visas i Figur 1. Tanken är att den befintliga datamodellen i framtiden kan utökas med klassen *Manning*. Ett sådant objekt representerar en bemanning som krävs för en viss typ av aktivitet, t.ex. en viss typ av personal som ska jobba ett visst antal timmar. I framtiden kan varje sådan bemanning även kräva en eller flera kompetenser.

Uppdragsgivaren vill undvika en onödigt komplicerad datastruktur i den händelse att systemet inte kommer att utökas med mer komplexa krav på modellen. Därför kommer till en början ett attribut på klassen *Label* att användas för att beskriva antalet bemannade timmar, istället för att utöka datamodellerna med en ny klass. Skapandet av en separat *Manning*-klass skulle i så fall kräva en mindre datamigrering.

Om det finns ett behov av att följa upp faktisk bemanning, kan detta lösas med ett nytt attribut för antal bemannade timmar på klassen *Entity*.

4.2 Användargränssnitt

Vi tror att det smidigaste alternativet är att kunna byta mellan olika lägen i användargränssnitten, beroende på om man vill visa planerad bemanning eller planerad produktion. En svårighet är att representera detta på ett bra sätt för användaren, då till synes likadana siffror då får radikalt olika betydelse beroende på i vilket läge man befinner sig. Därav kommer framtagningen av gränssnittet ske iterativt, där olika versioner kan utvärderas i samarbete med systemets användare.

En central del är den totala summan av antalet bemannade timmar. Detta tillhandahålls i användargränssnittet som en möjlighet att se summeringar för timmarna. Det vore troligen lämpligt att kunna summera dessa såväl totalt och per veckodag som per aktivitet.