Abstract (engelska)

Sammandrag (svenska)

Introduktion

Bakgrund

Något som liknar det vi skrev i planeringsrapporten.

Rapportens syfte

Rapportens syfte är att beskriva utvecklingen av läromaterialet, läromaterietl i sig samt den tekniska bakgrund som krävs för att förstå.

Projektets mål

Projekets mål är att skapa ett läromaterial som kombinerar fysik med tillhörande domänspecika språk. DSL:erna ska modellera utvalda fysikaliska områden och den tillhörande brödtexten ska förklara både DSL:erna i sig, fysik i sig samt kopplingen mellan dem.

Läromaterialet ska i slutändan bestå av en hemsida. Även källkoden till allt ska finnas tillgängligt.

Teori / teknisk bakgrund

Ska vi som de gjorde tidigare, förklara Git, Haskell, DSL, och fysik?

Haskell och funktionell programmering

I funktionell progammering uttrycker man allting i små och självständiga funktioner. Rekursion används ofta. Fördelen med detta är att programmen blir koncisa och de tenderar sakna de "progammeringsteknsika" delar som behövs för att få progammet att fungera, men som inte tillför någon betydelse till det man uttrycker.

Haskell är ett funktionellt progammeringsspråk med ett starkt typsystem. Ett exempel nedan

```
fakultet :: (Num n) => n -> n
fakultet 0 = 1
fakultet n = n * fakultet $ n-1        *
```

Domänspecifika språk

Ett domänspecifikt språk är som namnet låter ett språk som är gjort till en viss domän. En domän kan vara t.ex. fysikaliska enhter, eller tillgångar och skulder i ett företag. Eftersom språket är specifikt för domänen kan saker i den uttryckas enklare än i ett generellt språk.

Det finns två kategorier av domänspecifika språk, fristående och inbäddade. Skillnaden är att ett fristående är ett nytt progammeringsspråk från grunden medan ett inbäddat är skapat i ett värdspråk, och använder det språkets syntax. I detta fall kommer inbäddade domänspecika språk att skapas i Haskell.

Ett typexempel på ett domänspecifkt språk är ett syntaxträd för algebraisk uttryck, här kodat i Haskell.

```
data Expr = Expr :+: Expr
            Expr :*: Expr           *
            Const Double
            VarX

exempel = (Const 7.0 :+: VarX) :*: ((VarX :+: Const 10.0) :*: VarX)
```

Ha med en bild på detta.

Fysik vi behanldar och Fysik för ingenjörer

Fysik för ingenjörer är en fysikkurs som är obligatorisk för Datateknik i 2:an. Det är en grundläggande fysikkurs som behandlar mekanik, termodynamik och vågrörelselära

Den innehåller även en hel del tillämpad matematik, exempelvis vektorer och differentialkalkyl. Det användas bland annat vid beräkning av värmeledning.

LHS, Pandoc, HTML-hemsidan

Metod

Skapandet av läromaterialet har i grova drag haft tre faser. Först valdes olika arbetsområden ut, som enskilt gick att arbeta med. Sedan Skapades läromaterial för dessa områden. Till sist sammanfoagdes resultatet.

Selektion av arbetsområden

För att hitta områden att arbeta med studerades främst kursboken *Univeristy Physics*. Där lästes de kapitel som ingick i *Fysik för ingenjörer*. Sådant som verkade hade syntax som behövde förklaras, eller sådant som var svårt, eller sådant som var spännande valdes ut. De områden som hittats sorterades upp i grupper som var så fristående som möjligt för att kunna arbetas med på parallellt.

De områden som valdes ut blev - Vektorer - Enheter - Momentan och genomsnitt - Differentalkalkyl

Skapande av de första områdena

Varje gruppmedlem fck varsitt område att arbeta med. Man började med att experimentera med DSL:et för att hitta bra sätt att representera området på, vad som var tydligt och lätthanterat i datorn.

Det skedde också en del Haskell-inläsning av nya områden, exempelvis typnivå-progammering, för att kunna göra DSL:er på bästa sätt.

När en tanke börjat formas så implementerades först DSL:et. När det till stora delar var klart började förklarande brödtext skrivas till det, främst för att förklara koden som skrivits.

När koden var färdig och kommenterad tillräckligt väl började brödtexten uppdaters för att även innehålla mer kopplingar till fysik.

Sammanfogning av flera områden

Stack, git kan säkert passa här för att beskriva hur vi samarbetade med sammanfogningen.

Skapandet av hemsidan

Resultat

Läromaterialet (i sig)

En kombination av brödtext och DSL:er. De två är sammanvävda där fysik eller relaterad matematik presenters för att sedan modelleras i DSL:er.

Områden som är behandlade är - Dimensioner - Vektorer - Analys - Och mer med tid att vi arbetar vidare...

Specifikt om DSL:erna

Avgränsade DSLer för separata områden. Modellerar olika saker snarare än som en uppgiftslösare.

Specifikt om brödtexten

Lättsamma stilen och de roliga bilderna.

Hemsidan

Något om källkoden i sig?

Diskussion

Metoddiskussion

Vi har kanske inte varit så strukturerade, utan bara valt ut något område på måfå som vi kände för, t.ex. valet av termodynmik.

Blev så för svårt att veta vad som DSL lämpligt för.

Resultatdiskussion

Vad för slags områden är DSLs lämpligt att göra för?

Analys känns som ett område där DSLs kan vara lämpligt. Likaså dimensioner och vektorer. Vad har dessa gemensamt?

Gemensamt för de ovanstående är att de är matematiska, har en fix struktur, rigoröst hur de fungerar och har "data och operationer".

DSL / data	Ex operationer
Dimensioner	Multiplikation, division
Vektorer	Skalärprodukt, vektorprodukt
Analys, funktioner	Derivera, multiplicera

Operationerna på datan resultar också i ny data av samma slag (t.ex. vektorprodukt av två vektorer ger en ny vektor)

Varför gör detta dem lämpliga att göra DSLs för? För datan har en fix form. Och operaion av data blir ny data som evalueras till någon av formerna.

Detta gör DSL till områdena lämpliga ur två synvinklar. Dels enkelt att göra det rent praktiskt i Haskell. Dels för att man med DSL:et strukuretat upp och tydliggjort all data och operationer som går att göra.

Skillnad på enkelt att göra rent tekniskt, och vad som är lämpliga DSLs för att underlätta förståelsen av något.

DSLs var svårare för lutande plan och termodynamik (och problemlösning i allmänhet). Vad har dessa gemensamt?

Gemensamt för dessa är att det finns teoretiska samband/ekvationer som relaterar olika egenskaer i problemet. T.ex. för det lutande planet `a = g * sin v` är sambandet mellan `a` och `v` (och `g` som dock är en konstant bara). För termodynamik är det att t.ex. samband mellan inre energi, antal atomer i gasen och temperaturen.

Visst kan man modellera dessa samband. Men vad för nytta gör det? Problemlösning handlar om att känna till vilka samband som finns och tillämpa dem på olika sätt beroende på uppgift.

Man kan ju programmera en ekvationslösare, men den måste vara väldigt mekanisk av sig. Gör det kanske inte enklare för en själv att lösa problem.

Och vad är skillnaden mellan dessa två kategorier av områden?

Den viktiga skillnaden är att t.ex. analys har tydlig data och operationer medan problemlösning som lutande plan har ett gäng samband som man använder beroende på behov.

Gör DSLs så att fysik blir enklare att förstå?

Ex med lutande plan: man kan betrakta ett sådant problem som en samling ekvationer. Man har några kända värden och med hjälp av ekvationerna ska man hitta den sökta obekanta. Hjälper verkligen ett DSL till att man blir bättre på detta?

Med enheter, räcker det inte att förklara skillnaden mellan storheter, dimensioner och enheter på ett så grundligt sätt vi gjorde, utan att blanda in DSLs? Problemet i fysik för Data kanske är att det inte förklras grundligt och mycket är underförsått. Behöver man ens förstå enheter grundligare för att klara fysik bättre?

Man kan också se det som att DSLs och denna extrakunskap vi presenterat är för att göra fysik intressant genom att visa på vad för kopplingar till programmering man kan göra, även om områdena vi behandlat inte är direkt de områden som man behöver förstå för att klara kursen. Projektet kan ses som ren kuriosa som kan vara intressant.

Slutsatser

Etik

Källförteckning

Bilagor