CHALMERS

Learn You a Physics for Great Good

Using domain spedific languages to learn physics

DAVID FRISK

Instutitionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2018

Rätt namn på "dep ment"?

Kandidatarbete DATX02-18-08

An Informative Headline describing the Content of the Report

A Subtitle that can be Very Much Longer if Necessary

Björn Werner
Erik Sjöström
Johan Johansson
Oskar Lundström

Sortera efter efter namn?

CHALMERS

Instutionen för Data- och Informationsteknik
Division of Division name
Chalmers tekniska högskola
Göteborg, Sverige 2018

An Informative Headline describing the Content of the Report
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME

Supervisor: Name, Company or Department
Examiner: Name, Department

Master's Thesis 2018:NN
Department of Some Subject or Technology
Division of Division name
Name of research group (if applicable)
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wind visualization constructed in Matlab showing a surface of constant wind speed along with streamlines of the flow.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2018

An Informative Headline describing the Content of the Report
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME
Department of Some Subject or Technology
Chalmers University of Technology

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: lorem, ipsum, dolor, sit, amet, consectetur, adipisicing, elit, sed, do.

Sammanfattning

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: lorem, ipsum, dolor, sit, amet, consectetur, adipisicing, elit, sed, do.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Name Familyname, Gothenburg, Month Year

Innehåll

Figurer

Tabeller

1

Introduktion

1.1 Bakgrund

Vi måste göra något med bakgrunden. Bland annat anpassa efter de kommentarer vi fick i PP.

På civilingenjörsprogrammet Datateknik på Chalmers ingår den obligatoriska fysikkursen *Fysik för ingenjörer*. Tentastatistiken för denna kurs är inte jättebra[1]. Vi tror att många studenter på datateknik ("datateknologer") finner denna kurs svår eller ointressant, och att detta leder till att ungefär en tredjedel av kursdeltagarna får underkänt på tentamen.

Examinatorn för kursen "Fysik för ingenjörer, TIF085 (2016)" Åke Fäldt, tycker att studenter i allmänhet verkar ha svårt för att sätta upp egna modeller. De baserar sina mentala modeller helt eller delvis på intuition och felaktiga antaganden, istället för definitioner och bevisade satser som man är säker på gäller. Detta leder till att de tar genvägar som ofta är fel.

Sedan våren 2016 har kursen "Domain Specific Languages of Mathematics" ("DSLsofMath") eller "Matematikens domänspecifika språk" givits som en valbar kurs på kandidatnivå för studenter på Chalmers och Göteborgs Universitet. År 2016 var Cezar Ionescu huvudföreläsare, och från 2017 är Patrik Jansson huvudföreläsare. Det direkta målet är att förbättra den matematiska utbildningen för datavetare och den datavetenskapliga utbildningen för matematiker, där den grundläggande idéen bakom kursen är:

> "[. . .] att uppmuntra studenterna att närma sig matematiska domäner från ett funktionellt programmeringsperspektiv: att ge beräkningsbevis (calculational proofs); att vara uppmärksamma på syntaxen för matematiska uttryck; och, slutligen, att organisera de resulterande funktionerna och typerna i domänspecifika språk."[2][3]

Konkret så presenterar kursen matematik så som derivator, komplexa tal och matriser ur ett funktionellt programmeringsperspektiv i det funktionella programmeringsspråket Haskell. Dessa för studenterna bekanta verktyg används för att lösa matematiska problem så som modellering av syntax, evaluering till semantiska värden och datorassisterad bevisföring.

Även på MIT har en kurs inte helt olik DSLsofMath tidigare givits som berör både fysik

och domänspecifika språk (”DSL”). “Classical Mechanics: A Computational Approach” gavs av Prof. Geral Sussman och Prof. Jack Wisdom bl.a. år 2008.[4] Denna kurs på avancerad nivå studerar de fundamentala principerna av klassisk mekanik med hjälp av beräkningsidéer för att precist formulera principerna av mekanik, med början i Lagranges ekvationer och avslut i perturbationsteori (teori för approximationer av matematiska lösningar). I kursen används boken “Structure and Interpretation of Classical Mechanics” av Sussman, Wisdom och Mayer, vilken förklarar fysikaliska fenomen genom att visa datorprogram för att simulera dem, skrivna i språket Scheme.[5]

Utöver DSLsofMath-kursen har det även tidigare gjorts ett kandidatarbete om DSL här på Chalmers. Vårterminen 2016 utfördes kandidatarbetet “Programmering som undervisningsverktyg för Transformer, signaler och system. Utvecklingen av läromaterialet TSS med DSL” av fem studenter från Datateknik och Teknisk Matematik på Chalmers. Arbetet bestod av utveckling av läromaterial med tillhörande programmeringskod, uppgifter och lösningar, som komplement till existerande kurser i signallära.[6]

Vår tanke med detta kandidatarbete är att, likt premissen bakom kursen DSLsofMath och kandidatarbetet från 2016, angripa fysik på ett sådant sätt att ämnet blir både intressant och roligt för datateknologer, och därmed förhoppningvis också enklare. Med hjälp av domänspecifika språk skrivna i Haskell för att modellera fysik, d.v.s. samma pedagogiska verktyg som används inom datakurser, tror vi att kopplingen mellan fysikkursen och datateknikprogrammet kan göras tydligare och lärandet kan underlättas. Förhoppningen är att bl.a. det kraftfulla typsystemet i Haskell ska hjälpa studenter att bygga mentala modeller som är korrekta och inte bygger på felaktig intuition och antaganden.

Projektet är relevant för datateknologer som läser en fysikkurs. Men det kan också bli relevant för en fysikstudent som är ute efter en inkörsport till funktionell programmering. Förhoppningsvis blir det också relevant för de som är intresserade av domänspecifika språk i stort, pedagoger och föreläsare inom de berörda områdena och kanske till och med programledningen som ser vår rapport som ett skäl att introducera innehåll av detta slag i till exempel fysikkursen.

För läsaren som inte är insatt i domänspecifika språk, kan det förklaras som ett språk konstruerat för ett specifikt område, en domän. Språket kan användas för att enklare uttrycka saker inom domänen, till exempel Newtons andra lag $F = m \cdot a$ än vad som är möjligt inom generella (programmerings) språk. För vidare läsning rekommenderas *DSL for the Uninitiated*.[7]

1.2 Projektets mål

Målet med projektet är att skapa domänspecifika språk för fysik samt ett tillhörande läromaterial. Läromaterialets syfte är att beskriva fysiken och dess koppling till de domänspecifika språken projektgruppen utvecklat. Förhoppningen är att väcka intresse för fysik hos datateknologer genom att presentera fysik från ett annat perspektiv.

Läromaterialet är menat att i slutändan bestå av en hemsida. Källkoden för projektet skall vara offentligt tillgänglig.

Det är inte uppenbart hur en kombination av fysik och domänspecifika språk ser ut. En del av projektets mål är därför att ta reda på hur en sådan kombination *kan* se ut, samt om det finns en pedagogisk nytta.

1.3 Avgränsningar

Svårt att veta vad som kommer hinnas med.

Läromaterialet begränsar sig till att enbart beskriva en del områden inom fysik som ingår i kursen Fysik för ingenjöer. Denna avgränsningen valdes dells för att det är den fysik gruppmedlemmarnas kunskaper begränsar sig till, samt att det är den nämda kursen detta projekt kan bli mest relevant för, då kursen är en del av datastudenternas obligatoriska utbildning.

Fysik för ingenjörer behandlar grunderna inom de tre områdena mekanik (inklusive stelkroppsmekanik), termodynamik och vågrörelselära. Vi har valt att i första hand prioritera mekanik, vilket även innefattar tillämpad matematik i form av Euklidisk geometri och infinitesimalkalkyl. Dessutom har de områden där datateknologer haft svårigheter prioriterats. I mån av tid har vi valt att behandla termodynamik och vågrörelselära (WE WILL SEE).

Läromaterialet kommer inte att testas av en testgrupp under projektets gång. Detta ty en sådan undersökning hade krävt omfattande tid, genomgång av återkoppling samt omformning av läromaterialet. Även om projekets direkta syfte inte är att förbättra tentastatistiken på *Fysik för ingenjörer*, är vår förhoppning att förståelsen för de svåra delarna ska bli bättre.

1.4 Rapportens syfte

Rapportens syfte är att beskriva utvecklingen av läromaterialet, läromaterialet i sig samt den tekniska bakgrund som krävs för att förstå läromaterialet. Vi går in på svårigheter projektgruppen stött på under utvecklingen av läromaterialet, vilka områden av fysik som lämpat sig väl samt lämpat sig mindre väl för implementering i ett domänspecifikt språk med syfte att tjäna som läromaterial.

2

Teori

2.1 Domänspecifika språk

Ett domänspecifikt programmeringsspråk är ett språk som är avgränsat till ett specifikt domän. Detta domän kan ta många former, det kan vara ett språk för att formatera text på en hemsida (HTML), det kan vara ett språk för att interagera med en databas (SQL), ett språk för ett beskriva hur karaktärer ser ut (typsnitt). Användningsområden för dessa språk är väldigt smala men detta smala fokus gör det möjligt att utveckla ett rikt och lättanvändligt språk för just detta område.

Motsatsen till ett domänspecifikt språk är ett generellt språk (C, Java, Python) som är turingkompletta, vilket betyder att det kan uttrycka alla beräkningsbara problem i dem och även lösa dem givet tillräckligt med tid och minnestillgångar. Begränsningen med dessa generella språk är just deras egen generaliserbarhet, eftersom de har stöd för alla typer av beräkningar så blir både läsbarheten och användarvänligheten lidande.

referns

Ett domänspecifikt språk kan antingen implementeras som ett fristående språk eller bäddas in i ett redan existerande språk. De domänspecifika språk som utvecklats inom detta projekt är inbäddade i språket *Haskell*.

VARFÖR BLIR DET ENKLARE?

2.2 Haskell och funktionell programmering

Haskell är ett funktionellt programmeringsspråk som lämpar sig bra för att implementera ett domänspecifikt språk i. Anledningen till detta är den lätthet som man kan skapa nya datatyper och klasser för att representera grundstenarna i det nya språk, och även dess mönstermatchning som gör det möjligt att på enkelt sätt bryta isär komplexa datatyper för evaluering.

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

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

Figur 2.1: Bild!?

2.3 Syntaxträd och deras evaluering

???Osäker på om detta avsnitt är nödvändigt???

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

I detta exempel visar hur uttrycket $(7 + x) * ((x + 10) * x)$ kodas.

Ha med en bild på detta.

!!!Plus förklaring om evaluering!

2.4 Fysik vi behandlar och Fysik för ingenjörer

Fysik för ingenjörer är en obligatorisk kurs för studenterna på det datatekniska programmet. Kursen täcker grundläggande fysikområden såsom mekanik, termodynamik och vågrörelselära. Det ingår även en del tillämpad matematik såsom vektorer och differentialkalkyl. Dessa områden ämnas att täckas in av de domänspecifika språken och det tillhörande läromaterialet.

2.5 Hemsidan

a beskrivning
v individuella
knologier

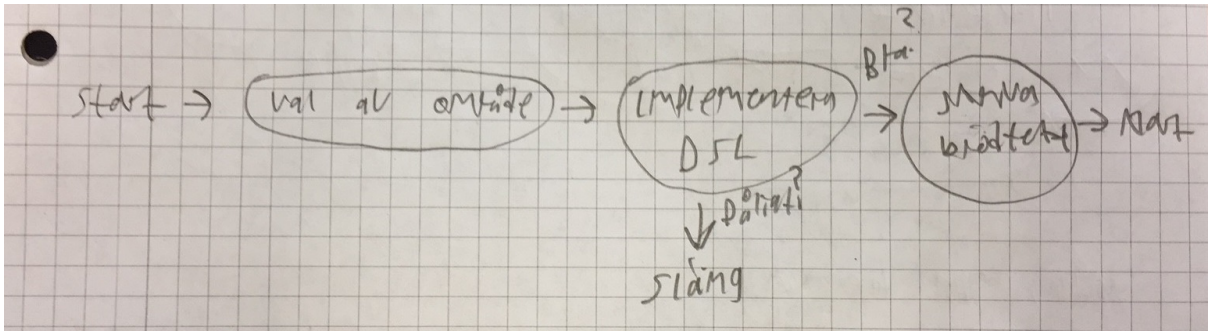
Programfilerna har skrivits i ett språk som kallas *Literate Kaskell* som kombinerar vanlig Haskell-kod med brödtext till något som kan tolkas av en kompilator som två separata saker. *Pandoc* är ett program som används för att generera html-filer med läromaterialet som är den slutgiltiga produkt som finns tillgänglig på hemsidan.

3

Metod/Genomförande: Beskrivning av konstruktionen av läromaterialet

TODO: Nånting om hur processen såg ut, lite mer meta. I.e. hur arbetade vi i allmänhet? Scrum, vattenfall? Med issue-tracker, med jira? Versionshantering? Etc.

Läromaterialet består av ett antal kapitel som vardera behandlar ett fysikaliskt område. Skapandet av läromaterialet har skett *kapitelvis*. För varje kapitel har en process följts. Denna process illustreras i figur 3.1.



Figur 3.1: Översikt över processen med att skapa kapitlena i läromaterialet.

Som figur 3.1 visar så bestod processen med att skapa ett kapitel av tre steg. Det första steget var att välja ett fysikaliskt område att behandla. Den andra steget var att implemenetera ett DSL för området. Det tredje steget var att skriva förklarande brödtext till det området.

Figuren visar också att, beroende på utfallet i implementationssteget, så antingen skrotas ett område eller gås vidare med. Huruvida området är lämpligt diskuteras i <ett diskussionskapitel>.

Läromaterialet publicerades i slutändan på en tillhörnade hemsida.

Resten av detta kapitel behandlar de olika stegen i den kapitelvisa processen mer i detalj. Det behandlar också publicering på hemsidan i närmare detalj.

TODO: Sektion om våra litteraturstudier? Känns lite skumt att ha, men 2016 hade. Litteratur är väl bara intressant att ha med som refens när man refererar till fakta eller

motiverar ett val? Själva studieprocessen, i.e. HUR man nådde beslutet, är väl inte alls lika intressant att ha med som VARFÖR man tog beslutet?

3.1 Selektion av arbetsområden

TODO: Hur hjälpte det senare Åke mötet? Med vad?

De områden som behandlas i läromaterialet kan klassificieras som antingen *grundläggande* eller *komposita*. Ett grundläggande område är ett område som inte bygger på något tidigare område. Ett exempel är vektorer. Ett komposit område är ett område som bygger på andra områden. Ett exempel är lutande plan, som använder sig av vektorer.

3.1.1 Grundläggande områden

Först kontaktades Åke Fäldt, som är examinator för kursen TIF085, Fysik för Ingenjörer. Åke befrågades om vilka områden han tycker att studenter verkar ha svårt för, och svarade att studenter i allmänhet verkar ha svårt för att sätta upp egna, mentala modeller för många problem och koncept. “Man tar genvägar (som ofta är fel) och bygger inte från det som man är säker på gäller” skrev han, och han pekade ut infinitesimalkalkyl som ett speciellt svårt område: “Infinitesimalkalkyl är ju en av hörnpelarna i fysiken och studenterna har trots att de har läst ganska mycket matematik jättesvårt att använda detta på verkliga system”.

För att identifiera mer specifika ämnesområden att arbeta med, studerades kursboken *Univeristy Physics*. Speciellt av intresse var kapitel som berörde saker som Åke Fäldt tidigare pekat ut som svåra, och kapitel som använde sig av specifik syntax. Domänspecifik syntax var av intresse att finna, då en betydlig del av domänspecifika språk är modellering av just syntaxen. Även områden som projektgruppen fann personligen intressanta, och områden som inte var av direkt intresse, men som utgjorde en kritisk beståndsdel av mer intressanta områden, valdes ut.

De valda grundområdena definierades sådana att de var fristående, i så god mån som möjligt, för att kunna arbetas med parallellt. De områden som valdes ut blev: Vektorer, Enheter, och Matematisk analys.

Vektorer eftersom det är en viktig grundsten i mekanik. Alla krafter, hastigheter, och accelerationer, betraktas oftast som vektorer i planet eller rymden, och dessa är alla fundamental element i mekanik.

Enheter eftersom det är viktigt för studenter att förstå sig på hur dimensioner påverkas av algebraiska operationer. Det kan också vara hjälpsamt att kunna utföra automatisk, datorassisterad dimensionsanalys på ens beräkningar.

Analys eftersom alla koncept i klassisk mekanik är relaterade genom matematisk analys. Mer specifikt används differenser för att beskriva medelrörelse, och infinitesimalkalkyl för att beskriva momentanrörelser. Vidare var infinitesimalkalkyl just det område som Åke Fäldt pekade ut som speciellt viktigt, och något som studenter har svårt för.

3.1.2 Komosita områden

När en mängd av de fristående, grundläggande områdena var färdigimplementerade så formulerades mer tillämpade områden i form av problemmängder. De tillämpade fysikproblemen krävde ofta bruk av flera olika DSL för att lösa. Exempelvis kan ett rörelseproblem använda sig av både differentialkalkyl och dimensionsanalys. Till dessa områden skrevs därför komposita DSL som kombinerade en mängd grundläggande DSL för att bättre kunna beskriva problemdomänen.

De tillämpade områdena var inte lika lätthanterliga som de grundläggande områdena. Det var sällan uppenbart hur komposita DSL skulle skrivas och appliceras, och vissa områden visade sig vara olämpade att skriva DSL för. Vidare motivation av varför vissa områden var mindre lämpade för den sortens DSL som skrevs finnes under Diskussionskapitlet. (TODO: Fast en kort motivation här med kanske?). Detta medförde att experimentering med DSL implementation var ett viktigt steg i processen att selektera lämpliga tillämpade områden, då experimenteringen kunde visa hurvida ett område var lämpat att skriva DSL för eller inte. TODO: oj vad hände med denna meningen, jag är trött.

De komposita områden som valdes ut blev: Momentan- och medel-rörelse, TODO: Mer, etc.

Momentan- och medel-rörelse eftersom de direkt utgör en stor delmängd av alla problem inom mekanik på den aktuella nivån. Väldigt många av de uppgifter studenter lär sig lösa inom mekanik är sträcka/hastighet/acceleration/kraft problem. Hur lång tid tar det att åka en sträcka om man har en viss medelhastighet? Om ett objekt med massa m påverkas av en kraft som varierar enligt $sin(t)$, vad är då momentanhastigheten vid $t = 10$?

TODO: Nåt om hur områdesvalen speglas i strukturen av läromaterialet?

3.2 Implementation av DSL för områdena

Det första steget vid implementering av ett DSL var alltid experimentering. Det finns ingen absolut, kanonisk metod att skriva ett DSL på, speciellt för specifik fysik, så experimentering var viktigt för att finna lämpliga representationer av syntax och andra element av DSL.

TODO: Förbättra

3.2.1 Implementation av

Av de områden som selekterats för projektet, valde varje gruppmedlem varsitt område att arbeta med. Från experimentering fanns syntaxträd och funktioner som väl kunde representerade matematiken och andra koncept inom området. Kraven för vad som ansågs som en bra representation var i huvudsak baserade på intuitionen hos gruppmedlemmarna som erfarna haskell-programmerare, och i viss mån individuellt definierade för varje gruppmedlem. I praktiken var de slutgiltiga implementationerna oftast också de minsta implementationerna, med avseende på bl.a. antal kodrader.

TODO: Stycket om "vår intuition som erfarna haskell-programmerare" lät lite pretentiöst. Annat ordval? "vår intuition som studenter som läst DSLsofMath kursen"?"

Alla områden krävde i varierande mån inläsning och studering av Haskell, Agda, fysik, matematik, och DSL. Till exempel krävde enhets-DSLet inläsning om ett gäng Haskell-extensions och typnivå-programmering. För att kunna bevisa korrekthet i beteende för vissa DSL studerades även Agda.

I allmänhet implementerades DSL i Haskell som en kombination av syntaxträd, funktioner för att manipulera dessa träd, och evalueringsfunktioner till någon slags semantisk domän. Komposita DSL var istället mer TODO: beskrivning. Varje DSL, grundläggande och komposit, parades även ihop med en mängd fysikproblem att appliceras på, sådant att läsaren ska få förståelse för hur DSLen kan brukas utöver hur de implementeras.

TODO: Vi bevisade/försökte bevisa grejer med. Skriv om att DSL i allmänhet hade bevis eller tester för att verifiera korrekthet.

Syntax analyserades och modellerades i syntaxträd. För vissa områden (enheter?) definierades även ett semantiskt värde som representerade en slags kanonisk form.

För vektorer gjordes ...

För enheter gjordes ...

För analys gjordes ...

För annat gjordes ...

3.2.2 Komposita områden

Importera DSLerna för varandra för att göra mer komplicerade grejer.

Eller, *Bruk av de mer fundamentala/teoretiska DSLerna för att angripa områden av mer "tillämpad" natur (såsom Krafter, Arbete, etc))*(?)

!! Områden/moduler soom bygger vidare på redan implementerade områden.

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

3.3 Skriva lärotext

En massa bra och fina didaktiska metoder användes för att skriva riktigt fin lärotext. Texten skrevs i LHS format, vilket innebar att beskrivande text alltid finns i samband med koden den har att göra med.

TODO: Referera till Donald knuth och literate programming. Literate programming som "paradigm" känns relevant för hur vi skriver lärotext.

TODO: Motivera varför vi valde LHS

TODO: Motivera varför vi valde hemsida istället för PDF.

TODO: Motivera varför vi valde Markdown istället för LaTeX i LHS filerna.

3.3.1 Didaktik/språk/utlärningsmetod

Lättsamt språk o en gnutta humor för att hålla kvar uppmärksamhet. Relaterat till Attention i ARCS modellen (2016 använde den. Såg vettig ut).

TODO: Expandera. Vad exakt använder vi för didaktisk metod? Samma som appliceras i Learn You a Haskell, mer eller mindre. Fånga uppmärksamhet med lite humor och lättsamhet etc.

3.3.2 Skriva lärotext till DSL

När en modul DSL var färdigimplementerad, ofta med nödtorftiga kodkommentarer, skrevs brödtext som förklarade kod-delen. Detta efterföljande steg för att förstå koden är viktigt, och är ett krav för att man ska kunna skriva en förklaring (och förstå den) om kopplingen till fysik.

Vidare skrevs brödtexten som sammankopplade DSL och fysik. När själva koden var på plats skrevs hur den relaterar till och modellera fysik. I slutändan skrevs också inledning och avslutning till kapitlet.

3.3.3 Skriva läromaterial för hur DSL appliceras för problemlösning

Vari vi visar att DSLerna både är praktiskt användbara, likt Wolfram Alpha, och att implementationen+applikationen hjälper oss förstå mekanik i allmänhet och probleminstanserna i synnerhet.

3.4 Sammanställning, presentation, och publicering

Läromaterialet publiceras på en internethemsida, varpå man kan läsa allt o ha skoj.

3.4.1 Beskrivning

Ett build-script hämtar .lhs källfilerna, i vilka brödtexten är skriven med markdown. Rendrar med pandoc, och sätter in lite navigationselement etc. med hjälp av eget templating-system. Manuellt läggs sedan stoffet på gh-pages branchen för att automatiskt visas på dslsofmath.github.io/BScProj2018.

Obs: Medan bygget är scriptat så är inte publiceringen det, och ingenting genereras/publiceras automatiskt kontinuerligt. Måste köra scriptet manuellt och lägga stoff på gh-pages branchen.

3.4.2 Build-script

I.e. implementation av python-build-scriptet i mer detalj.

TODO: Är detta ens intressant? Viktigt för att producera sidan såklart, men inte intressant ur varken matte eller haskell/DSL perspektiv.

3.4.3 Hemsidan

Nåt om design, läslighet, grafik(?), navigation, avsiktligt undvikande av javascript, etc.

Tänker lite samma med denna sektion som ovan. Har ju ingenting med varken matte eller DSL att göra i sig, så kanske inte så intressant? Samtidigt är det kanske lite intressant ur pedagogik-aspekten. Kan det kanske vara lättare/roligare att lära sig om sidan är fin och lättläst? Att javascript inte krävs gör att sidan kan visas ordentligt även om man sitter i U-land med dålig/gammal/billig telefon.

3.5 Test och återkoppling

Test på försöksstudenter. Återkoppling med Åke(?).

Nog bra att vara explicit här med att det inte är en rigorös empirisk studio, om inte det redan täckts väl i Avgränsningar.

4

Metod2/Genomförande2: Skapandet av läromaterialet

Arbetet med att skapa läromaterialet har haft ett antal delvis parallella faser. Den första fasen (1) var att skapa läromaterial till separata områden inom fysik. Den andra fasen (2a) var att kombinera de tidigare DSLerna till komposita områden. Den tredje fasen (2b) var att använda tidigare DSLer och tillämpa på fysikaliska problem. Den fjärde fasen (2c) var att publicera läromaterial på en hemsida.

Den första fasen var tvunget att ske innan någon annan, då de andra beror på den första. Men efter att lite olika fristående DSLer skapats kunde de tre andra faserna börja ske parallellt. Men den första fasen fortsattae. Olika medlemmar i gruppen arbetade med olika faser samtidigt.

4.1 Fas 1: Läromaterial för separata områden

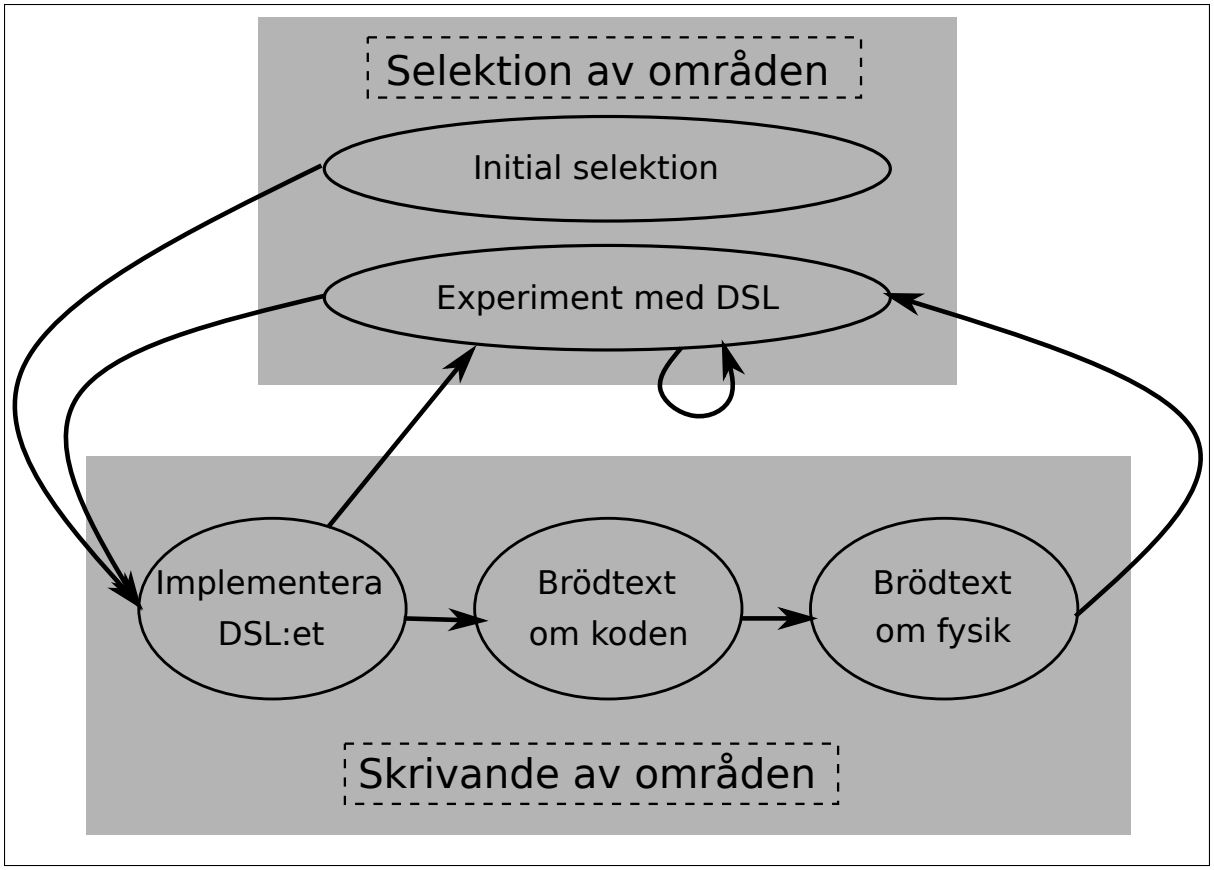
Processen med att skapa läromaterial för separata områden har varit en loopande process. 4.1 visar en översikt över processen med att skapa läromaterial för de separata områden. Som figuren visar har processen itererat fram och tillbaka mellan de två huvudsakliga stegen selektion och skrivande.

Snygga till figuren

4.1.1 Selektion av områden

Att hitta områden att arbeta med inom fysik var inget trivialt att göra. De främsta källorna till inspiration var kursboken och annat material som tillhörde kursen. Ett område skulle vara fristående från andra områden, vara grundläggande samt vara ”lämpligt”.

Vad för slags områden detta var, var inget som gick att se bara genom att titta på det. Istället fick det experimenteras med området för att upptäcka om det var ”lämpligt”, dvs gick att representera på ett lätthanterligt och bra sätt i datorn. Dessa krav specifiefas



Figur 4.1: Översikt över processen med att skapa de separata områdena.

närmare i resultat-kapitelt (då vi visste vad dessa karaktäristika var).

I samband med experiment med DSL skedde också en del inläsning, av bland annat Haskell och Agda. Ett exempel var typnivå-programmering. Detta gjordes för att kunna göra DSL:et på bästa sätt.

Långt ifrån alla försök med områden blev lyckade. Märktes att ett område inte skulle bli något bra DSL så började vi om processen med att hitta ett nytt område. Som tidigare nämnt, så står det vad som var ett bra område och vad som var ett dåligt senare i resultat-kapitlet.

Sökandet efter ett område har utförsts flera gånger i projeket. Ofta skedde det individuellt då. Ett speciellt fall av sökande är dock den initiala selektionen, som skedde i början av projektfasen. Då lästes hela kursboken och allt innehåll radades upp och sorterades. På detta sätt kunde några initiala områden väljas ut att arbteras med. Dessa områden blev

- Vektorer
- Dimensioner
- Momentan och genomsnitt
- Differentialkalkyl

4.1.2 Skrivande av områden

När ett område verkat fruktsamt och en tanke om hur ett DSL kan se ut börjat formats, så påbörjades en ordentlig implementation av DSL:et. DSLet implementerades i en nästan fullständig utsträckning innan något annat på området gjordes, för att vara säkra att området verklig var lämpligt att göra. Ibland visade det sig, trots de inledande experimenten indikerrade så, att området inte var bra. Då skrotades området.

När hela DSLet var implementerat, ofta med nödtorftiga kodkommentarer, började brödtext som förklarade kod-delen skrivas. Detta var det efterföljande steget för att förstå koden är viktigt, och är ett krav för att man ska kunna skriva en förklaring (och förstå den) om kopplingen till fysik.

Det sista steget är att skriva brödtexten som gör kopplingen mellan DSL och fysik. När själva koden var på plats skrevs hur den relaterar till och modellera fysik. I slutändan skrevs också inledning och avslutning till kapitlet.

4.2 Fas 2a: Läromaterial för komposita områden

Importera DSLerna för varandra för att göra mer komplicerade grejer.

Eller, *Bruk av de mer fundamentala/teoretiska DSLerna för att angripa områden av mer "tillämpad" natur (såsom Krafter, Arbete, etc))*(?)

!! Områden/moduler soom bygger vidare på redan implementerade områden.

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

Selektionen gick till som...

Processen med att skriva läromaterialet gick till på samma sätt som för de separata områdena. Se de kapitlet.

Här har vi inte gjort så mycket är eller?

4.3 Fas 2b: Tillämpning av DSL:er för problemområden

Vari vi visar att DSLerna både är praktiskt användbara, likt Wolfram Alpha, och att implementationen+applikationen hjälper oss förstå mekanik i allmänhet och probleminstanserna i synnerhet.

Fritt fall, lutande plan...

Selektion genom...

Skrivande på samma sätt som för separata områden.

4.4 Fas 2c: Publicering på hemsidan

Först en inledande konstruktion av hemsidan.

Sedan löpande att material kompileras och läggs där.

5

Resultat: Beskrivning av det resulterande läromaterial

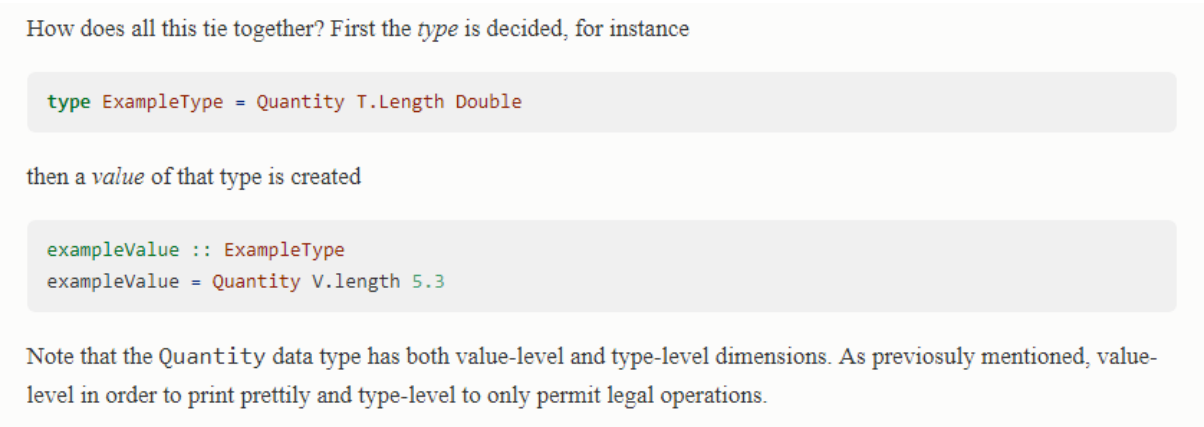
5.1 Lärotexten i sig

Läromaterialet blev i slutändan, precis som målet var, en sammanvävning mellan domänspecifika språk som modellerar fysik och brödtext som förklarar både fysiken i sig, men också förklarar de domänspecifika språken.

Figur 5.1 visar ett smakprov över det resulterande läromaterialet. Som figuren visar är brödtext sammanvävt med Haskell-kod för domänspecifika språk.

Läromatateriale behandlar ett flertal områden inom fysik, samt matematik som används inom fysik. Fokuset är på mekanik samt till det området tillhörande matematik. I sin fullständighet är de behandlade områdena

- Analys
- Bevis
- Dimensioner
- Fysikaliska kroppar
- Vektorer



How does all this tie together? First the *type* is decided, for instance

```
type ExampleType = Quantity T.Length Double
```

then a *value* of that type is created

```
exampleValue :: ExampleType
exampleValue = Quantity V.length 5.3
```

Note that the `Quantity` data type has both value-level and type-level dimensions. As previosuly mentioned, value-level in order to print prettily and type-level to only permit legal operations.

Figur 5.1: Ett smakprov över hur det resulterande läromaterielt ser ut.

Analys handlar om matematisk analys och bygger upp ett syntaxträd... TODO: kolla mer detaljrikt.

I *Bevis*-kapitlet presenteras bevisföringen med hjälp av Haskells typsystem. Det exemplifieras genom att kinematiska formler bevisas.

Dimensioner behandlar dimensioner, storheter och enheter inom fysiken. Dimensioner införs på typnivå i Haskell för att kunna visa på likheten mellan Haskells typsystem och hur man måste förhålla sig till dimensioner inom fysiken.

Fysikaliska kroppar vet ej TODO: ta reda på

Vektorer vet ej TODO: ta reda på

I läromatateriale finns, förutom de fem ovanstående separata områdena, även tillämpningar av de områdena på exempelproblem. Till exempel används *Dimensioner* till att lösa problem med fritt fall. TODO: fyll på när vet mer om hur tillämpningarna ser ut.

5.1.1 Domänspecifika språk

De domänspecika språken är avgränsade och behandlar separata områden. De är avgränsade för att det är enklare att förstå dem om de är det. Av samma skäl behandlar de separata områden. Om det skulle uppstå behov kan de kombineras istället.

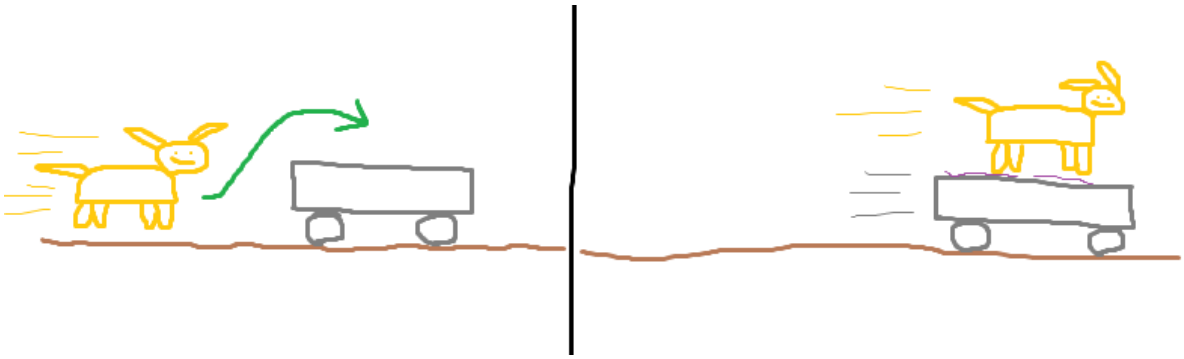
De domänspecifika språken modellerar områden snarare än att vara problemlösare. *Analys* exemplifierar detta väl. Det språket består av ett syntaxträd över algebraiska uttryck samt operationer som derivering och integration. Med hjälp av det kan man modellera uttryck och analytiska operationer på dem. Däremot löser det inte problem åt en. Man kan med andra ord inte mata in en differentialekvation och automatiskt få en lösning.

TODO: Detta kanske passar bättre i diskussion: Hur bra våra moduler blev samt varför de inte är problemlösare utan modellerare istället.

5.1.2 Brödtexten

Brödtexten finns till som ett förklarande komplement till de domänspecifika språken. Texten förklarar dels den bakomliggande fysiken, dels den Haskell-kod som finns. Generellt står förklaring av kod för en stor del av brödtexten. Detta för att det är själva modellerandet som är den stora delen - det är egentligen ganska lite fysik som presenteras. Därav behöver koden en utförlig förklaring. Dessutom används avancerade koncept i Haskell, bland annat typnivå-programmering, som läsaren inte förväntas kunna sedan innan. Av naturliga skäl kräver dessa en längre förklaring.

Texten är skriven på engelska för att komma fler till gagn än om den varit skriven på



Figur 5.2: Exempel på bild ur läromaterilet

svenska.

Språket i texten är vardagligt och lättsamt. Detta för att vara en kontrast mot hur kursböcker vanligtvis ser ut.

I brödtexten finns bilder. De har en rolig och medvetet kladdig stil. Syftet är att muntra upp läsaren. Figur **??** är ett exempel på en bild som finns i läromaterielet. Den visar den humoristiska och oseriösa ritningstekniken.

Valet att skriva på engelska, den lättsamma stilen och de roliga bilderna har inspirerats av *Learn You a Haskell*. TODO: referens. Läromaterialet är tänkt att vara något liknanden, men för fysik istället för Haskell.

TODO: Blev det en bra text? Kanske ska vara i diskussionen istället

5.2 Hemsidan

Läromaterialet finns tillgängligt på en hemsida, som består av grundläggande HTML, CSS och javascript. På hemsidan finns en innehållsförteckning med klickbara länkar till de olika kapitlen. Hemsidan är öppen för alla och bör fungera i de flesta webläsare. Javascript är inget krav för hemsidan. Matematiska formler visas ändå, om än inte lika tydligt.

5.3 Källkod

En del av projektets mål var att all källkod skulle finnas fritt tillgänglig. Det gör den också. Det finns tillgänglig på internet på projektes GitHub-repository TODO: referens.

Källkoden som finns är all den som använts under projekets gång, både slutversion och alla mellanversioner sedan projekets början. Källkoden är inte bara läromaterialet i sig, utan även kod till hemsidan, rapporter (inklusive denna) och mötesprotokoll.

5.4 Återkoppling från testgrupp

För att utvärdera huruvida läromaterialet är intressant och hjälpsamt har vi haft en informell återkoppling med en testgrupp. TODO: Skriv mer när vi faktiskt gjort detta.

6

Diskussion

6.1 Tillvägagångssättet av skapandet

Tanke: I metod/teori varför vi valde att just Haskell o.s.v. Beskrivning av vårt iterativa arbetssätt, hur vi sökte i mörkret. Här i diskussion kanske mer varför det var nödvändigt...

Annan tanke: antingen separat resultat-kapitel för misslyckadeförsök, som DSL för lutande plan, eller att vi har det här..

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.

Största svårigheten var att välja områden som skulla passa bra för DSL. Som att famla i mörkret.

Kan skriva om misslyckade försök att skapa DSLer för vissa saker. T.ex. lutande plan (DSL för det, ej problemlösning av det området med andra DSL:er), bevis med Haskells typsystem...

6.2 Domänspecifika språk och fysik

6.2.1 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".

Orkar inte göra Latex tabell... | 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.

6.2.2 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.

7

Slutsatser

Beskriv vad vi kom fram till i diskussionen. Varför är det svårt med vissa områden. Tips till dem som vill testa något liknande.

8

Etik

blablabla

Litteraturförteckning

[1] G. Johansson, "Statistik över kursresultat", 2018. [Online]. Tillgänglig: http://document.chalmers.se/doc/00000000-0000-0000-0000-00001C968DC6, hämtad: 2018-01-30.

[2] C. Ionescu och P. Jansson, "Domain Specific Languages of Mathematics: Lecture Notes", 2018. [Online]. Tillgänglig: https://github.com/DSLsofMath/DSLsofMath/tree/master/L/snapshots, hämtad: 2018-01-30.

[3] C. Ionescu och P. Jansson, "Domain-Specific Languages of Mathematics: Presenting Mathematical Analysis Using Functional Programming", i *Proceedings of the 4th and 5th International Workshop on Trends in Functional Programming in Education*, Sophia-Antipolis, France and University of Maryland College Park, USA, 2015, ss. 1-15. [Online]. Tillgänglig: https://doi.org/10.4204/EPTCS.230.1, hämtad: 2018-01-30.

[4] G. J. Sussman och J. Wisdom, "Classical Mechanics: A Computational Approach", 2008. [Online]. Tillgänglig: urlhttps://ocw.mit.edu/courses/earth-atmospheric-and-planetary-sciences/12-620j-classical-mechanics-a-computational-approach-fall-2008/, hämtad: 2018-01-30.

[5] G. J. Sussman och J. Wisdom, *Structure and Interpretation of Classical Mechanics*. Cambridge, MA, USA: MIT, 2001. [Online]. Tillgänglig: https://mitpress.mit.edu/sites/default/files/titles/content/sicm/book.html, hämtad: 2018-03-07.

[6] F. Lindahl, C. Rosvall, P. Ngo, J. Jonsson och J. Olsson, "Programmering som undervisningsverktyg för Transformer, signaler och system: Utveckling av läromaterialet TSS med DSL", Chalmers tekniska högskola, Göteborg, Sverige, 2016.

[7] D. Ghosh, "DSL for the Uninitiated", *Queue*, vol. 9, nr. 6, s. 10-21, jun. 2011. [Online]. Tillgänglig: https://doi.org/10.1145/1989748.1989750, hämtad: 2018-03-07.

A

Bilaga λ

Inläsning

- Identifikation av problemområden.
  - Kontakt med Åke Fäldt och DNS. Studera kursutvärderingar.
  - Reflektera över vad vi själva tyckt varit svåra områden då vi läst kursen.
- Studerande av existerande läromaterial, både inom ren fysik och liknande vårt material.
  - Fysikboken.
  - Åke Fäldts egna material.
  - Boken *Structure Interpretaton of Classical Mechanics*[5].
  - Kursboken till kursen *Matematikens domänspecifika språk*.
- Existerande implementationer.
  - OpenTA.
  - Hamilton.
  - MasteringPhysics.
- Tidigare forskning.
  - Cezar och Patriks 2015 forskningsartikel.
  - 2016 års kandidatarbete.
  - Artikeln *DSL for the Uninitiated*.
  - *Communicating Mathematics: Useful Ideas from Computer Science*

Implementation av domänspecifika språk

Vid implementationen av ett/flera domänspecika språk behöver nedanstående punkter genomföras.

- Hitta relevanta grundtyper inom fysik, exempelvis sträcka och massa.
- Hitta relevanta kompositttyper, exempelvis hastighet och tryck.
- Utförligt typsystem.
- Dimensionskontroll.
- Modellera fysikens syntax i språket.

- Pedagogiska syntaxträd.
- Kombinatorer och konstruktorer.
- Hålla våra typer polymorfa.

Skrivande av läromaterial

Vid skrivandet av läromaterialet kommer följande punkter ligga till grund.

- En gemensam vokabulär som fungerar när man skriver om både fysik och programmering (generics kontra polymorfism), och som gör det möjligt att prata om dem i samma mening utan att byta språk och på så sätt brygga det semantiska gapet mellan områdena.
- Övningar
  - Modellera ett fysikaliskt problem med vårt domänspecifika språk.
  - Lös ett ”vanligt” fysikaliskt problem med hjälp av vårt domänspecifika språk.
  - Simuleringar i stil med *Bouncing Balls*.
  - Delar av fysiken vi inte behandlat lämnas som övning att själv implementera.
- Gå igenom allmän teori (t.ex. Newtons lagar, krafter som verkar, etc) tillsammans med en parallell utveckling av ett domänspecifikt språk.
- Materialet ska vara enkelt att ta till sig.
- Verkligen exponera det DSL som vi gemensamt bygger för att påvisa kopplingen mellan fysik och programmering.