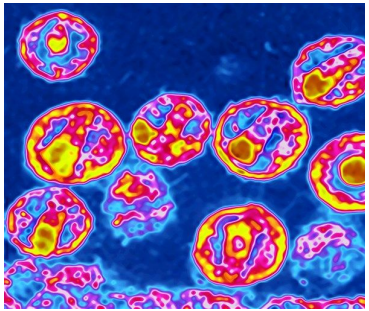
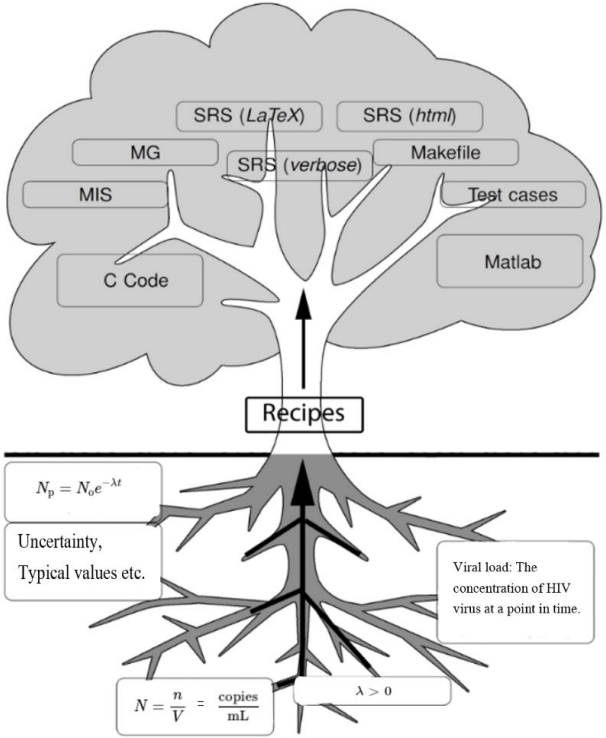


Drasil Demonstration for *Diagnosis*

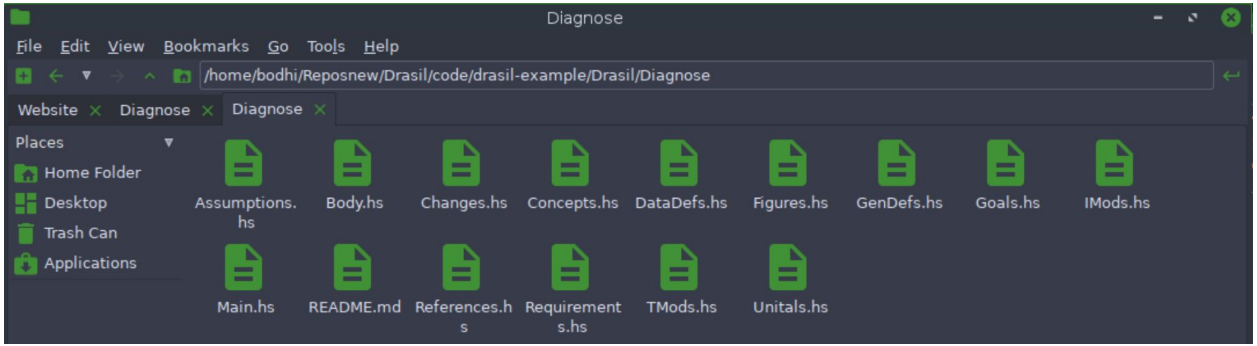
Andrea Clemeno

DRASIL FRAMEWORK AND DIAGNOSIS

- Framework that generates software artifacts and code.



DIAGNOSIS CODE OVERVIEW

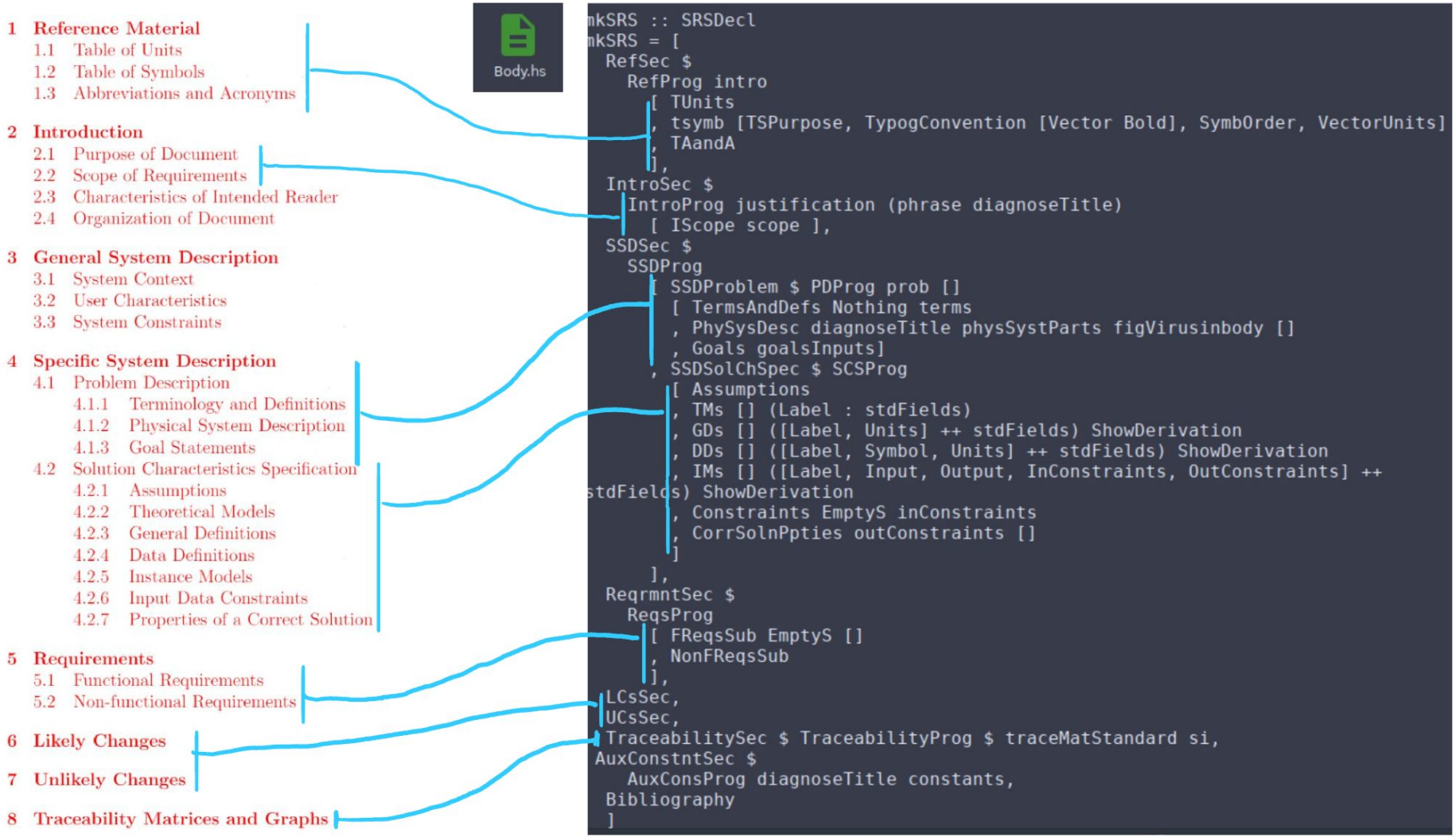


README.md

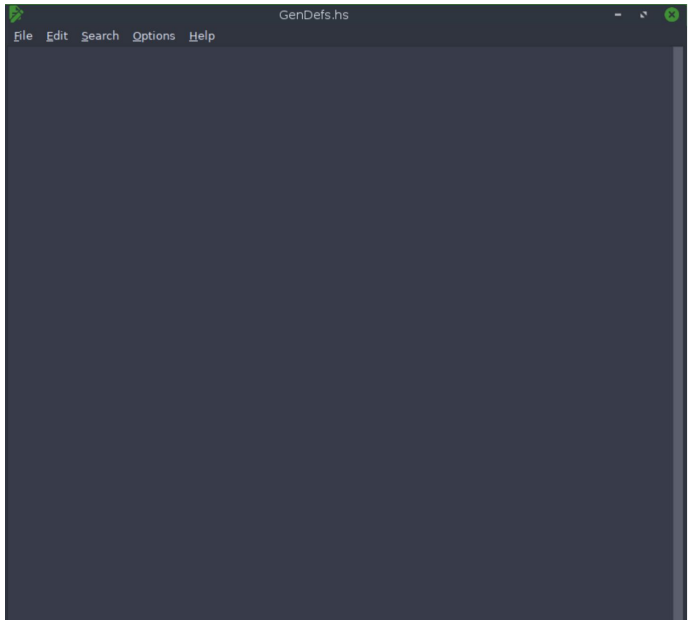
File Edit Search Options Help

```
Assumptions.hs
- The assumptions for Diagnose

Body.hs
- The main document body representation for the Diagnose example

Changes.hs
- The likely and unlikely changes for Diagnose

Concepts.hs
- The example-specific concepts for Diagnose

DataDefs.hs
- The data definitions for Diagnose

Figures.hs
- The figure for Diagnose

GenDefs.hs
- The general definitions for Diagnose
```

```
Goals.hs
- The goal statements for Diagnose

IMods.hs
- The instance models for Diagnose

Main.hs
- A list of what should be generated

README.md
- This file

References.hs
- The references for Diagnose

Requirements.hs
- The functional and nonfunctional changes for Diagnose

TMods.hs
- The theoretical models for Diagnose

Unitals.hs
- The example-specific quantities for Diagnose
```


1 Reference Material
1.1 Table of Units
1.2 Table of Symbols
1.3 Abbreviations and Acronyms
2 Introduction
2.1 Purpose of Document
2.2 Scope of Requirements
2.3 Characteristics of Intended Reader
2.4 Organization of Document
3 General System Description
3.1 System Context
3.2 User Characteristics
3.3 System Constraints
4 Specific System Description
4.1 Problem Description
4.1.1 Terminology and Definitions
4.1.2 Physical System Description
4.1.3 Goal Statements
4.2 Solution Characteristics Specification
4.2.1 Assumptions
4.2.2 Theoretical Models
4.2.3 General Definitions
4.2.4 Data Definitions
4.2.5 Instance Models
4.2.6 Input Data Constraints
4.2.7 Properties of a Correct Solution
5 Requirements
5.1 Functional Requirements
5.2 Non-functional Requirements
6 Likely Changes
7 Unlikely Changes
8 Traceability Matrices and Graphs
Body.hs
mkSRS :: SRSDecl
mkSRS = [
RefSec $
RefProg intro
[TUnits
, tsymb [TSPurpose, TypogConvention [Vector Bold], SymbOrder, VectorUnits]
, TAandA
],
IntroSec $
IntroProg justification (phrase diagnoseTitle)
[IScope scope],
SSDSec $
SSDProg
[SSDProblem $ PDProg prob []
[TermsAndDefs Nothing terms
, PhySysDesc diagnoseTitle physSystParts figVirusinbody []
, Goals goalsInputs]
, SSDSolChSpec $ SCSProg
[Assumptions
, TMs [] (Label : stdFields)
, GDs [] ([Label, Units] ++ stdFields) ShowDerivation
, DDs [] ([Label, Symbol, Units] ++ stdFields) ShowDerivation
, IMs [] ([Label, Input, Output, InConstraints, OutConstraints] ++
stdFields) ShowDerivation
, Constraints EmptyS inConstraints
, CorrSolnPpties outConstraints []
]
],
ReqrmntSec $
ReqsProg
[FReqsSub EmptyS []
, NonFReqsSub
],
LCsSec,
UCsSec,
TraceabilitySec $ TraceabilityProg $ traceMatStandard si,
AuxConstntSec $
AuxConsProg diagnoseTitle constants,
Bibliography
]

Generating the General Definition Section



General Definitions

This section collects the laws and equations that will be used to build the instance models.

Refname	**GD:vLoadt**
Label	VLoadt as a function of time for constant decay rate
Units	$\frac{\text{copies}}{\text{mL}}$
Equation	$N_t = N_o e^{-\lambda t}$
Description	N_t is the viral load at time t ($\frac{\text{copies}}{\text{mL}}$) N_o is the initial viral load ($\frac{\text{copies}}{\text{mL}}$) λ is the elimination constant (s^{-1}) t is the time (s)
Source	hobbie1970
RefBy	IM: calofPredictedVL and IM: calofElimConst

Detailed derivation of viral load at time t:

Using the First-Order rate Law in TM: expElim, we have:

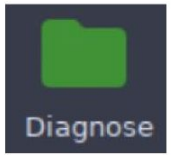
$$\frac{dN}{dt} = -\lambda N_o$$

Where N_t denotes the viral load at time t, N_o denotes the initial viral load and λ denotes the elimination constant . When rearranging for integration, we have:

$$\int_{N_o}^{N_t} 1 \, dN_t = -\int_0^t \lambda \, dt$$

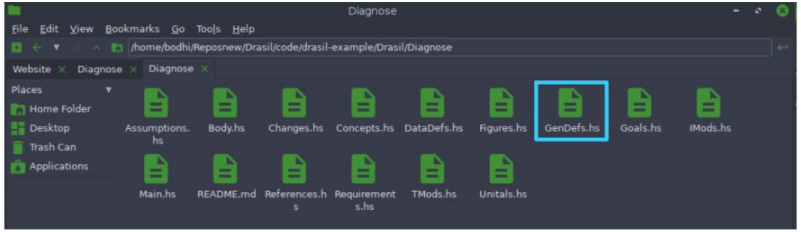
Performing the integration, we have the required equation:

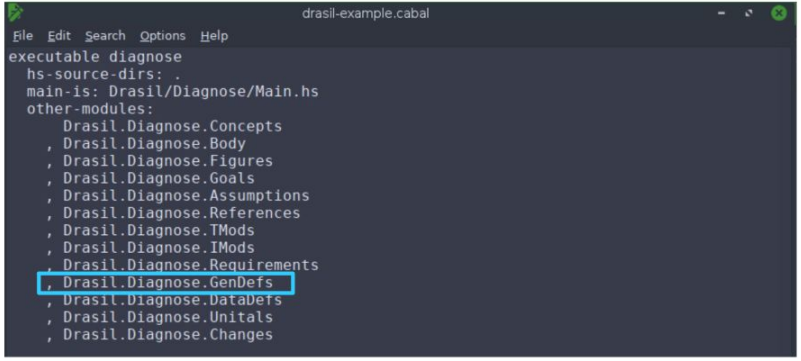
$$N_t = N_o e^{-\lambda t}$$

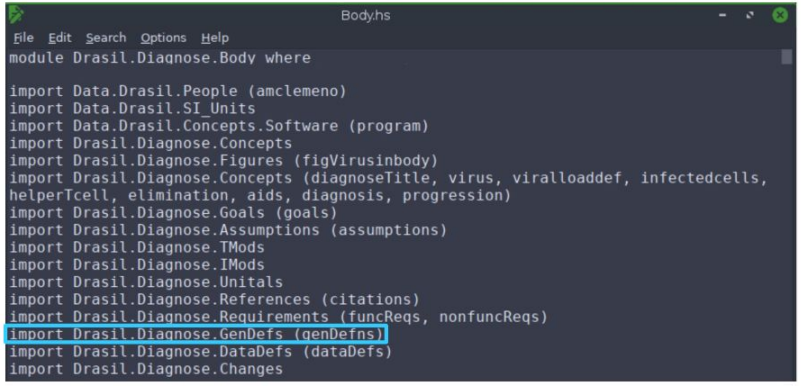

Diagnose


drasil-
example.cab
al


Body.hs


Diagnose
File Edit View Bookmarks Go Tools Help
/home/bodhi/Reposnew/Drasil/code/drasil-example/Drasil/Diagnose
Website
Diagnose
Diagnose
Places
Home Folder
Desktop
Trash Can
Applications
Assumptions.hs
Body.hs
Changes.hs
Concepts.hs
DataDefs.hs
Figures.hs
GenDefs.hs
Goals.hs
IMods.hs
Main.hs
README.md
References.hs
Requirements.hs
TMods.hs
Unitals.hs


drasil-example.cabal
File Edit Search Options Help
executable diagnose
hs-source-dirs: .
main-is: Drasil/Diagnose/Main.hs
other-modules:
Drasil.Diagnose.Concepts
, Drasil.Diagnose.Body
, Drasil.Diagnose.Figures
, Drasil.Diagnose.Goals
, Drasil.Diagnose.Assumptions
, Drasil.Diagnose.References
, Drasil.Diagnose.TMods
, Drasil.Diagnose.IMods
, Drasil.Diagnose.Requirements
, Drasil.Diagnose.GenDefs
, Drasil.Diagnose.DataDefs
, Drasil.Diagnose.Unitals
, Drasil.Diagnose.Changes


Body.hs
File Edit Search Options Help
module Drasil.Diagnose.Body where
import Data.Drasil.People (amclemeno)
import Data.Drasil.SI_Units
import Data.Drasil.Concepts.Software (program)
import Drasil.Diagnose.Concepts
import Drasil.Diagnose.Figures (figVirusinbody)
import Drasil.Diagnose.Concepts (diagnoseTitle, virus, viralloaddef, infectedcells,
helperTcell, elimination, aids, diagnosis, progression)
import Drasil.Diagnose.Goals (goals)
import Drasil.Diagnose.Assumptions (assumptions)
import Drasil.Diagnose.TMods
import Drasil.Diagnose.IMods
import Drasil.Diagnose.Unitals
import Drasil.Diagnose.References (citations)
import Drasil.Diagnose.Requirements (funcReqs, nonfuncReqs)
import Drasil.Diagnose.GenDefs (genDefns)
import Drasil.Diagnose.DataDefs (dataDefs)
import Drasil.Diagnose.Changes


```
module Drasil.Template.Body where

import Language.Drasil
import Language.Drasil.Printers (PrintingInformation(..),
defaultConfiguration)
import Database.Drasil (Block, ChunkDB, ReferenceDB, SystemInformation(SI),
  cdb, rdb, refdb, _authors, _purpose, _concepts, _constants, _constraints,
  _datadefs, _configFiles, _definitions, _defSequence, _inputs, _kind,
_outputs,
  _quants, _sys, _sysinfodb, _usedinfodb)
import Theory.Drasil (DataDefinition, GenDefn, InstanceModel, TheoryModel)
import Utils.Drasil

import Drasil.DocLang (SRSDecl, mkDoc)

import qualified Data.Drasil.Concepts.Documentation as Doc (srs)

srs :: Document
srs = mkDoc mkSRS (for'' titleize phrase) si

printSetting :: PrintingInformation
printSetting = PI symbMap Equational defaultConfiguration

mkSRS :: SRSDecl
mkSRS = []
```

```
symbMap :: ChunkDB
symbMap = cdb ([] :: [QuantityDict]) [nw example] ([] :: [ConceptChunk])
  ([] :: [UnitDefn]) ([] :: [DataDefinition]) ([] :: [InstanceModel])
  ([] :: [GenDefn]) ([] :: [TheoryModel]) ([] :: [ConceptInstance])
  ([] :: [Section]) ([] :: [LabelledContent])
```

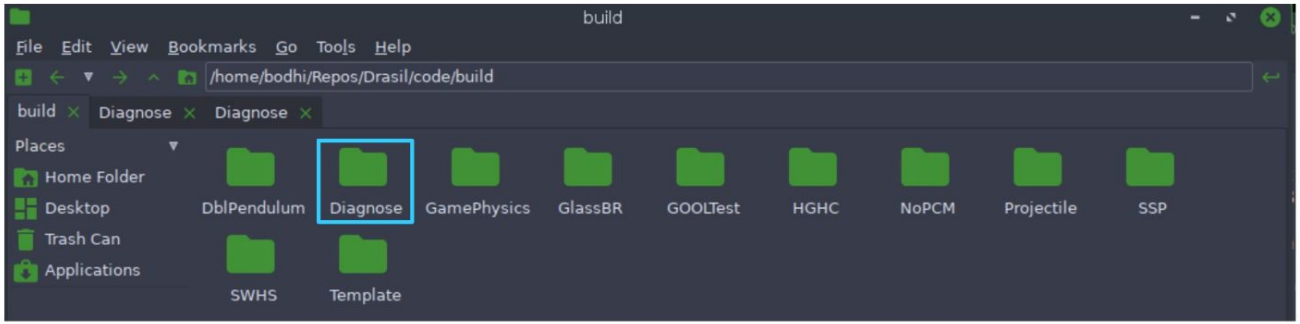

```
module Drasil.Diagnose.Body where
mkSRS :: SRSDecl
mkSRS = [
 RefSec $
   RefProg intro
     [ TUnits
     , tsymb [TSPurpose, TypogConvention [Vector Bold], SymbOrder, VectorUnits]
     , TAandA
     ],
 IntroSec $
   IntroProg justification (phrase diagnoseTitle)
     [ IScope scope ],
 SSDSec $
   SSDProg
     [ SSDProblem $ PDProg prob []
       [ TermsAndDefs Nothing terms
       , PhySysDesc diagnoseTitle physSystParts figVirusinbody []
       , Goals goalsInputs]
     , SSDSolChSpec $ SCSProg
       [ Assumptions
       , TMs [] (Label : stdFields)
       , GDs [] ([Label, Units] ++ stdFields) ShowDerivation
       , DDs [] ([Label, Symbol, Units] ++ stdFields) ShowDerivation
       , IMs [] ([Label, Input, Output, InConstraints, OutConstraints] ++
stdFields) ShowDerivation
       , Constraints EmptyS inConstraints
       , CorrSolnPpties outConstraints []
       ]
     ],
 ReqrmntSec $
   ReqsProg
     [ FReqsSub EmptyS []
     , NonFReqsSub
     ],
 LCsSec,
 UCsSec,
 TraceabilitySec $ TraceabilityProg $ traceMatStandard si,
AuxConstntSec $
   AuxConsProg diagnoseTitle constants,
 Bibliography
 ]
```

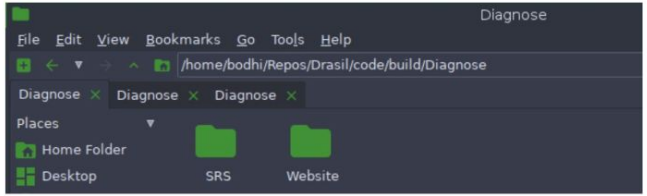
```
symbMap :: ChunkDB
symbMap = cdb (map qw physicscon ++ symbols) (nw diagnoseTitle : nw mass : nw
inValue : [nw program] ++
    map nw doccon ++ map nw doccon' ++ map nw physicalcon ++ map nw physicCon ++
map nw physicCon'++ map nw acronyms ++
    map nw mathcon ++ map nw fundamentals ++ map nw derived ++ map nw tMCC) (map cw
defSymbols ++ srsDomains)
  (siUnits) (dataDefs) (iMods)
  (genDefns) (tMods) (concIns)
  ([] :: [Section]) ([] :: [LabelledContent])
```

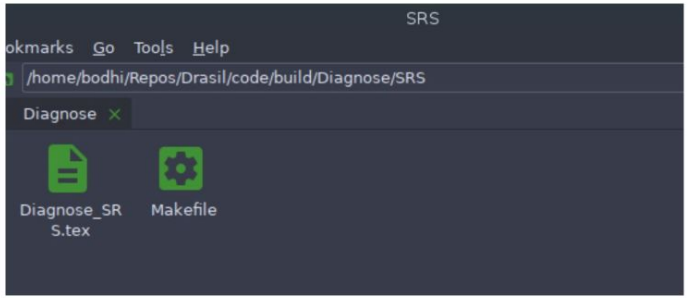

```haskell
module Drasil.Diagnose.GenDefs (genDefns, vLoadtGD) where

import Prelude hiding (exp)
import Language.Drasil
import Theory.Drasil (GenDefn, TheoryModel, gd, gdNoRefs)
import Utils.Drasil

import Drasil.Diagnose.Assumptions
import Drasil.Diagnose.TMods (expElimTM)
import Drasil.Diagnose.Unitals
import Drasil.Diagnose.References

genDefns :: [GenDefn]
genDefns = [vLoadtGD]

----------
vLoadtGD :: GenDefn
vLoadtGD = gd vLoadtRC (getUnit vLoad) (Just vLoadtDeriv)
  [makeCite hobbie1970] "vLoadt" [{-Notes-}]


vLoadtRC :: RelationConcept
vLoadtRC = makeRC "vLoadtRC" (nounPhraseSent $ foldlSent_
             [S "VLoadt as a function" `sOf` phrase time, S "for constant decay
rate"])
             EmptyS $ sy vLoadt $= sy vLoado * exp (negate(sy elimConst * sy time))

vLoadtDeriv :: Derivation
vLoadtDeriv = mkDerivName (phrase vLoadt) (weave [vLoadtDerivSents, map E
vLoadtDerivEqns])

vLoadtDerivSents :: [Sentence]
vLoadtDerivSents = [vLoadtDerivSent1, vLoadtDerivSent2, vLoadtDerivSent3]

vLoadtDerivSent1, vLoadtDerivSent2, vLoadtDerivSent3 :: Sentence

vLoadtDerivSent1 = foldlSentCol [S "Using the First-Order rate Law" `sIn` makeRef2S
expElimTM `sC`
                                 S "we have" ]

vLoadtDerivSent2 = foldlSentCol [S "Where", ch vLoadt +:+ S "denotes the", phrase
vLoadt `sC`
                                   ch vLoado +:+ S "denotes the", phrase vLoado
`sAnd` ch elimConst +:+
                                   S "denotes the", phrase elimConst +:+
                                   S ". When rearranging for integration" `sC` S "
we have"]

vLoadtDerivSent3 = foldlSentCol [S "Performing the integration" `sC` S "we have the
required equation"]


vLoadtDerivEqns :: [Expr]
vLoadtDerivEqns = [vLoadtDerivEqn1, vLoadtDerivEqn2, vLoadtDerivEqn3]

vLoadtDerivEqn1, vLoadtDerivEqn2, vLoadtDerivEqn3 :: Expr
vLoadtDerivEqn1 = deriv (sy vLoad) time $= negate(sy elimConst * sy vLoado)
vLoadtDerivEqn2 = defint (eqSymb vLoadt) (sy vLoado) (sy vLoadt) 1 $=
                  negate (defint (eqSymb time) 0 (sy time) (sy elimConst))
vLoadtDerivEqn3 = sy vLoadt $= sy vLoado * exp (negate(sy elimConst * sy time))
```

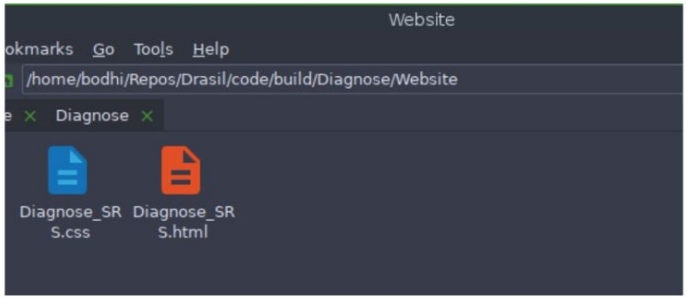
```
bodhi@bodhi-VirtualBox: ~/Repos/Drasil/code
bodhi@bodhi-VirtualBox:~$ cd Repos
bodhi@bodhi-VirtualBox:~/Repos$ cd Drasil
bodhi@bodhi-VirtualBox:~/Repos/Drasil$ cd code
bodhi@bodhi-VirtualBox:~/Repos/Drasil/code$ make
```

```
bodhi@bodhi-VirtualBox: ~/Repos/Drasil/code
diff --strip-trailing-cr -r -X ../.gitignore -x '*.txt' "stable/ssp/" "build/SSP
"/ > "logs/SSP_log.log" 2>&1
cd "build/GamePhysics" && stack exec -- "gamephysics"
diff --strip-trailing-cr -r -X ../.gitignore -x '*.txt' "stable/gamephysics/" "b
uild/GamePhysics"/ > "logs/GamePhysics_log.log" 2>&1
cd "build/Template" && stack exec -- "template"
diff --strip-trailing-cr -r -X ../.gitignore -x '*.txt' "stable/template/" "buil
d/Template"/ > "logs/Template_log.log" 2>&1
cd "build/DblPendulum" && stack exec -- "dblpendulum"
diff --strip-trailing-cr -r -X ../.gitignore -x '*.txt' "stable/dblpendulum/" "b
uild/DblPendulum"/ > "logs/DblPendulum_log.log" 2>&1
cd "build/Diagnose" && stack exec -- "diagnose"
diff --strip-trailing-cr -r -X ../.gitignore -x '*.txt' "stable/diagnose/" "buil
d/Diagnose"/ > "logs/Diagnose_log.log" 2>&1
cd "build/GOOLTest" && stack exec -- "codegenTest"
diff --strip-trailing-cr -r -X ../.gitignore -x '*.txt' "stable/gooltest/" "buil
d/GOOLTest"/ > "logs/GOOLTest_log.log" 2>&1
-----------------------------
Make complete, checking logs
-----------------------------

-----------------------------------------
- GENERATED OUTPUT MATCHES STABLE VERSION -
-----------------------------------------
bodhi@bodhi-VirtualBox:~/Repos/Drasil/code$
```


build
File Edit View Bookmarks Go Tools Help
/home/bodhi/Repos/Drasil/code/build
build
Diagnose
Diagnose
Places
Home Folder
Desktop
Trash Can
Applications
DblPendulum
Diagnose
GamePhysics
GlassBR
GOOLTest
HGHC
NoPCM
Projectile
SSP
SWHS
Template


Diagnose
File Edit View Bookmarks Go Tools Help
/home/bodhi/Repos/Drasil/code/build/Diagnose
Diagnose
Diagnose
Diagnose
Places
Home Folder
Desktop
SRS
Website


SRS
okmarks Go Tools Help
/home/bodhi/Repos/Drasil/code/build/Diagnose/SRS
Diagnose
Diagnose_SRS.tex
Makefile


Website
okmarks Go Tools Help
/home/bodhi/Repos/Drasil/code/build/Diagnose/Website
Diagnose
Diagnose_SRS.css
Diagnose_SRS.html

GenDefs.hs

```
GenDefs.hs
File  Edit  Search  Options  Help
module Drasil.Diagnose.GenDefs (genDefns, vLoadtGD) where

import Prelude hiding (exp)
import Language.Drasil
import Theory.Drasil (GenDefn, TheoryModel, gd, gdNoRefs)
import Utils.Drasil

import Drasil.Diagnose.Assumptions
import Drasil.Diagnose.TMods (expElimTM)
import Drasil.Diagnose.Unitals
import Drasil.Diagnose.References

genDefns :: [GenDefn]
genDefns = [vLoadtGD]

----------
vLoadtGD :: GenDefn
vLoadtGD = gd vLoadtRC (getUnit vLoad) (Just vLoadtDeriv)
  [makeCite hobbie1970] "vLoadt" [{-Notes-}]


vLoadtRC :: RelationConcept
vLoadtRC = makeRC "vLoadtRC" (nounPhraseSent $ foldlSent_
           [S "VLoadt as a function" `sOf` phrase time, S "for constant decay
rate"])
           EmptyS $ sy vLoadt $= sy vLoado * exp (negate(sy elimConst * sy time))

vLoadtDeriv :: Derivation
vLoadtDeriv = mkDerivName (phrase vLoadt) (weave [vLoadtDerivSents, map E
vLoadtDerivEqns])

vLoadtDerivSents :: [Sentence]
vLoadtDerivSents = [vLoadtDerivSent1, vLoadtDerivSent2, vLoadtDerivSent3]

vLoadtDerivSent1, vLoadtDerivSent2, vLoadtDerivSent3 :: Sentence

vLoadtDerivSent1 = foldlSentCol [S "Using the First-Order rate Law" `sIn` makeRef2S
expElimTM `sC`
                                 S "we have" ]

vLoadtDerivSent2 = foldlSentCol [S "Where", ch vLoadt +:+ S "denotes the", phrase
vLoadt `sC`
                                 ch vLoado +:+ S "denotes the", phrase vLoado
`sAnd` ch elimConst +:+
                                 S "denotes the", phrase elimConst +:+
                                 S ". When rearranging for integration" `sC` S "
we have"]

vLoadtDerivSent3 = foldlSentCol [S "Performing the integration" `sC` S "we have the
required equation"]


vLoadtDerivEqns :: [Expr]
vLoadtDerivEqns = [vLoadtDerivEqn1, vLoadtDerivEqn2, vLoadtDerivEqn3]

vLoadtDerivEqn1, vLoadtDerivEqn2, vLoadtDerivEqn3 :: Expr
vLoadtDerivEqn1 = deriv (sy vLoad) time $= negate(sy elimConst * sy vLoado)
vLoadtDerivEqn2 = defint (eqSymb vLoadt) (sy vLoado) (sy vLoadt) 1 $=
                  negate (defint (eqSymb time) 0 (sy time) (sy elimConst))
vLoadtDerivEqn3 = sy vLoadt $= sy vLoado * exp (negate(sy elimConst * sy time))
```

Diagnose_SRS.html

General Definitions

This section collects the laws and equations that will be used to build the instance models.

Refname	GD:vLoadt
Label	VLoadt as a function of time for constant decay rate
Units	$\frac{\text{copies}}{\text{mL}}$
Equation	$N_t = N_o e^{-\lambda t}$
Description	N_t is the viral load at time t ($\frac{\text{copies}}{\text{mL}}$) N_o is the initial viral load ($\frac{\text{copies}}{\text{mL}}$) λ is the elimination constant (s^{-1}) t is the time (s)
Source	hobbie1970
RefBy	IM: calofPredictedVL and IM: calofElimConst

Detailed derivation of viral load at time t:

Using the First-Order rate Law in TM: expElim, we have:

$$\frac{dN}{dt} = -\lambda N_o$$

Where N_t denotes the viral load at time t, N_o denotes the initial viral load and λ denotes the elimination constant. When rearranging for integration, we have:

$$\int_{N_o}^{N_t} 1\, dN_t = -\int_0^t \lambda\, dt$$

Performing the integration, we have the required equation:

$$N_t = N_o e^{-\lambda t}$$

Diagnose_SRS.tex

Diagnose_SRS.tex

File Edit Search Options Help

```
\\ \midrule \\
RefBy & \hyperref[GD:vLoadt]{GD: vLoadt}

\\ \bottomrule
\end{tabular}
\end{minipage}
\subsubsection{General Definitions}
\label{Sec:GDs}
This section collects the laws and equations that will be used to build the
instance models.

\vspace{\baselineskip}
\noindent
\begin{minipage}{\textwidth}
\begin{tabular}{>{\raggedright}p{0.13\textwidth}>{\raggedright\arraybackslash}
p{0.82\textwidth}}
\toprule \textbf{Refname} & \textbf{GD:vLoadt}
\phantomsection
\label{GD:vLoadt}
\\ \midrule \\
Label & VLoadt as a function of time for constant decay rate

\\ \midrule \\
Units & $\frac{\text{copies}}{\text{mL}}$

\\ \midrule \\
Equation & \begin{displaymath}
           {N_{\text{t}}}={N_{\text{o}}} e^{-λ t}
           \end{displaymath}
\\ \midrule \\
Description & \begin{symbDescription}
              \item{${N_{\text{t}}}$ is the viral load at time t ($
\frac{\text{copies}}{\text{mL}}$)}
              \item{${N_{\text{o}}}$ is the initial viral load ($
\frac{\text{copies}}{\text{mL}}$)}
              \item{$λ$ is the elimination constant ($\text{s}^{-1}$)}
              \item{$t$ is the time (${\text{s}}$)}
              \end{symbDescription}
\\ \midrule \\
Source & \cite{hobbie1970}
```

Diagnose_SRS.html

General Definitions

This section collects the laws and equations that will be used to build the instance models.

Refname	**GD:vLoadt**
Label	VLoadt as a function of time for constant decay rate
Units	$\frac{\text{copies}}{\text{mL}}$
Equation	$N_t = N_o e^{-\lambda t}$
Description	N_t is the viral load at time t ($\frac{\text{copies}}{\text{mL}}$) N_o is the initial viral load ($\frac{\text{copies}}{\text{mL}}$) λ is the elimination constant (s^{-1}) t is the time (s)
Source	hobbie1970
RefBy	IM: calofPredictedVL and IM: calofElimConst

Detailed derivation of viral load at time t:

Using the First-Order rate Law in TM: expElim, we have:

$$\frac{dN}{dt} = -\lambda N_o$$

Where N_t denotes the viral load at time t, N_o denotes the initial viral load and λ denotes the elimination constant . When rearranging for integration, we have:

$$\int_{N_o}^{N_t} 1 \, dN_t = -\int_0^t \lambda \, dt$$

Performing the integration, we have the required equation:

$$N_t = N_o e^{-\lambda t}$$

NEXT STEPS

Code generation:

1. Please refer to noPCM, or GlassBR to generate code or your project from Drasil
2. NoPCM example can be referred to, if your project requires an ODE, otherwise, you can refer to GlassBR example.
3. Navigate to C:\Drasil\code\drasil-example<project example you want to mirror either GlassBR or NoPCM>\Main.hs
4. Copy the content of Main.hs file and paste in your Main.hs project file. Keep in mind there are some elements in the file that is already in your main.hs file, so carefully delete duplicated or unwanted texts and modify other texts to suit your project.
5. Open Main.hs and modify the file to suit your project.
6. Modify your design choices
7. Update MakeFile.hs in code folder
8. Run 'make'
9. Your build folder should contain a src folder

Benefits	Challenges
- Very easy to change fundamental parts of your SRS. ie. units - Made my SRS more organized. - After repeating the process and knowing what the errors mean, it was very simple to add the knowledge of my project.	- Learning curve: Significantly harder in the beginning compared to final stages of generation. - Errors sometimes hard to navigate. ie parse errors, indentation errors etc.

Thank you!

References

[1] https://github.com/andreamclemeno/CAS741-Concentration-of-Virus/blob/master/docs/SRS/SRS.pdf

[2] https://jacquescarette.github.io/Drasil/
[3]
https://github.com/JacquesCarette/Drasil/tree/97b0fceceb522488b05ca1a2fdb12d0de1f889a8/code/stable/projectile/Projectile_C_P_NoL_B_U_V_D/src/python