Sustainable Software Product Lines via Generation

Spencer Smith and Jacques Carette

Computing and Software Department
Faculty of Engineering
McMaster University

Huawei 2023 Visit: Feb?, 2023

Generate All Things with Drasil

- **Goal** — Improve software sustainability and productivity
- **Ideas**
  - Adapt a software product line approach
  - Build on success of MDSE
  - Value all documents, not just code
  - Start with well-understood domains
- **Solution**
  - Capture (codify) knowledge once
  - Generate all documentation and code
  - Idealized dev process for well-understood software
- **Implement Partial Solution — Drasil**

Product Lines

Product Lines in User Manual

PRODUCT SPECIFICATIONS

The appearance and specifications listed in this manual may vary due to constant product improvements.

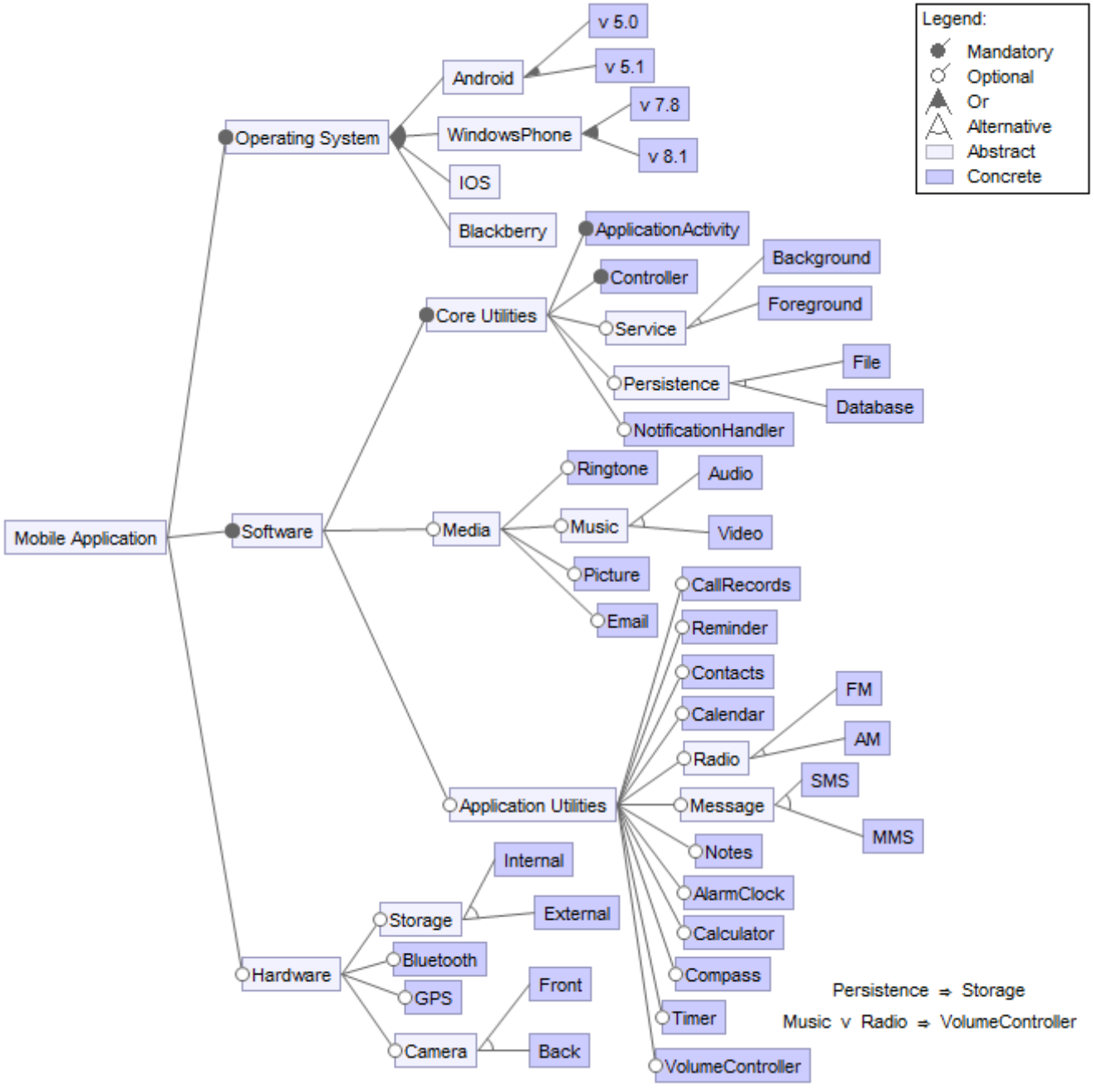
Electrical requirements: 115 V, 60 Hz

Min. / Max. water pressure: 20 - 120 psi (138 - 827 kPa)

Model	LFCC22426*
Description	Counter-depth, French door refrigerator, bottom freezer
Net weight	243 lb (110 kg)

Model	LFCS27596*
Description	Standard-depth, Door-in-Door French door refrigerator, bottom freezer
Net weight	284 lb (129 kg)

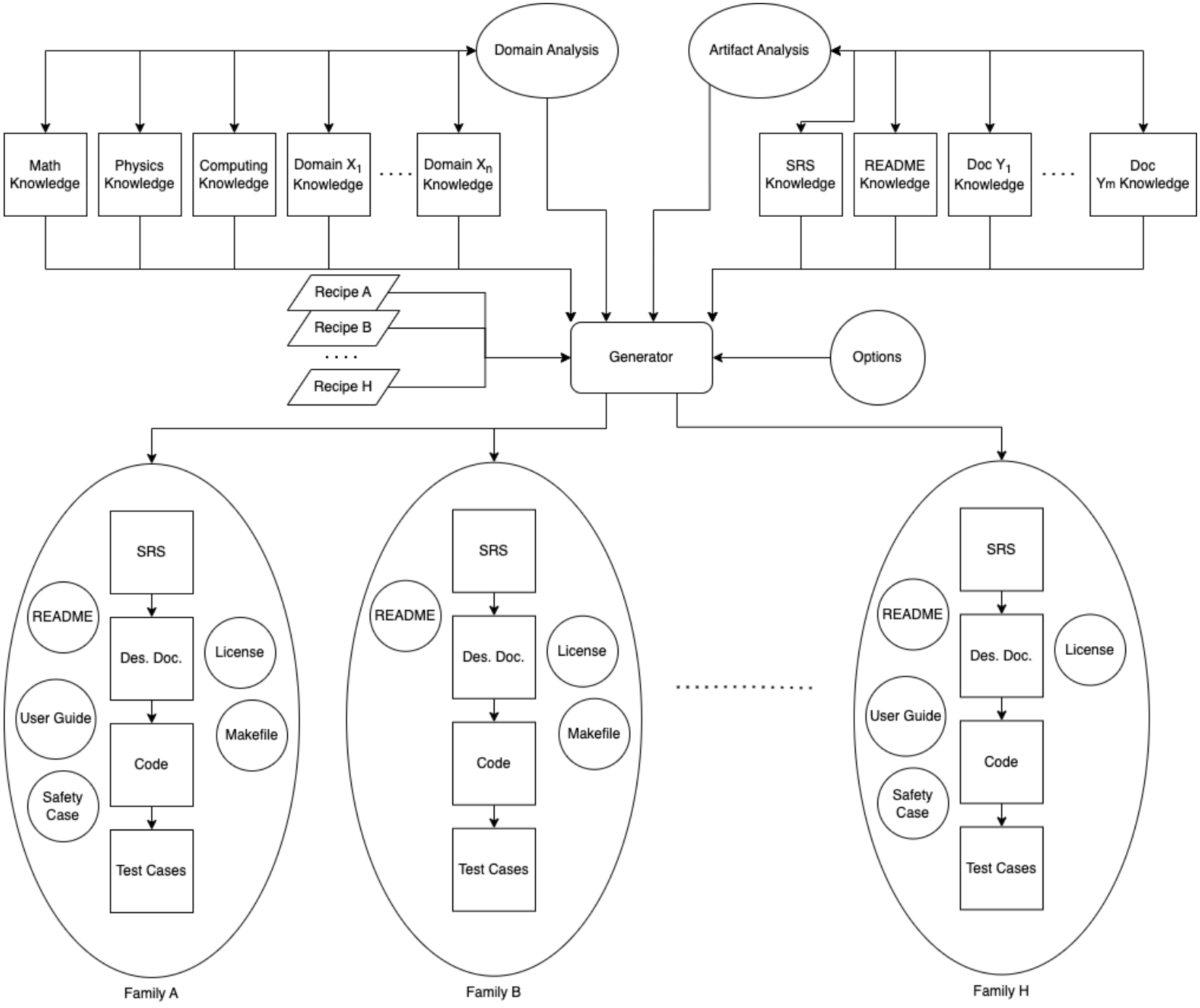
Model	LFCC23596*
Description	Counter-depth, Door-in-Door French door refrigerator, bottom freezer
Net weight	269 lb (122 kg)



Usman et al. (2017)

Build on Success of MDSE

- Codify (capture) code and non-code info together
  - Natural language (text)
  - Definitions
  - Assumptions
  - Rationale, Derivations
  - Abstract theory
  - Etc.
- Generate all artifacts from one framework
  - Requirements
  - User manuals
  - Build scripts, dev environment (CI etc)
  - Assurance case
  - Code (in different languages)
  - Test cases
  - etc.


Domain Analysis
Artifact Analysis
Math Knowledge
Physics Knowledge
Computing Knowledge
Domain X_1 Knowledge
Domain X_n Knowledge
SRS Knowledge
README Knowledge
Doc Y_1 Knowledge
Doc Y_m Knowledge
Recipe A
Recipe B
Recipe H
Generator
Options
SRS
README
Des. Doc.
License
User Guide
Makefile
Code
Safety Case
Test Cases
Family A
Family B
Family H

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} \text{ and } \mathbf{v} = \frac{d\mathbf{p}}{dt}$$

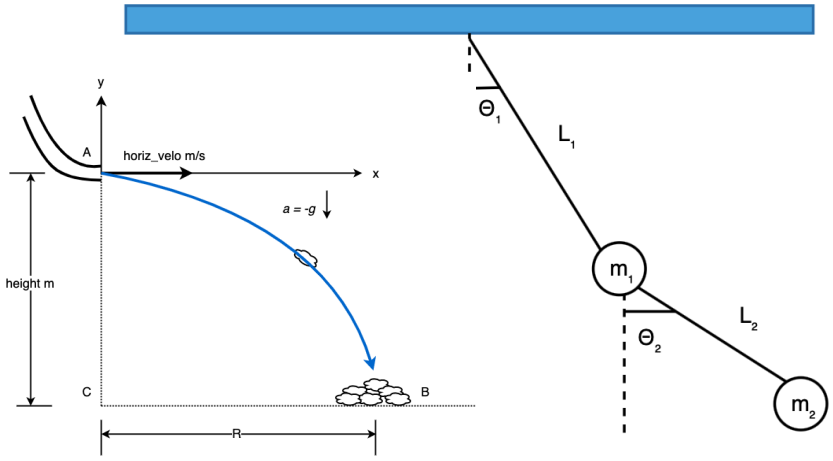
$$\mathbf{F} = m\mathbf{a}$$

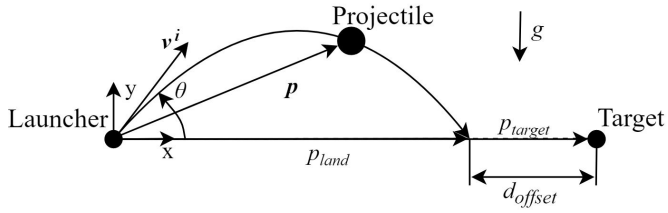
$$m\frac{d\mathbf{v}}{dt} = m\mathbf{g} - c\mathbf{v}$$

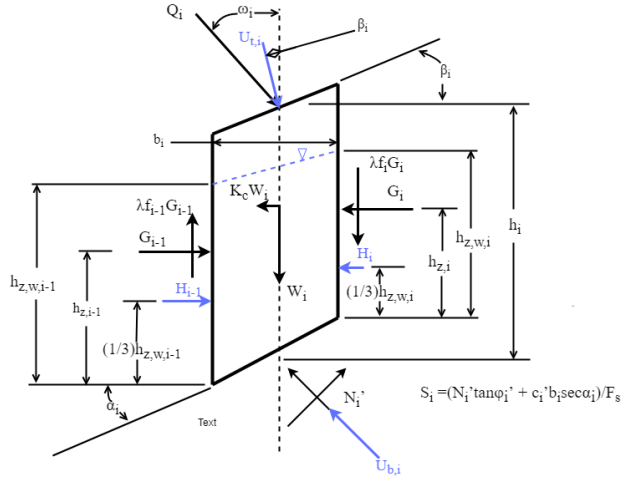
$$g = 9.8 m/s^2 \text{ or } g = 32.2 ft/s^2$$

$$= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

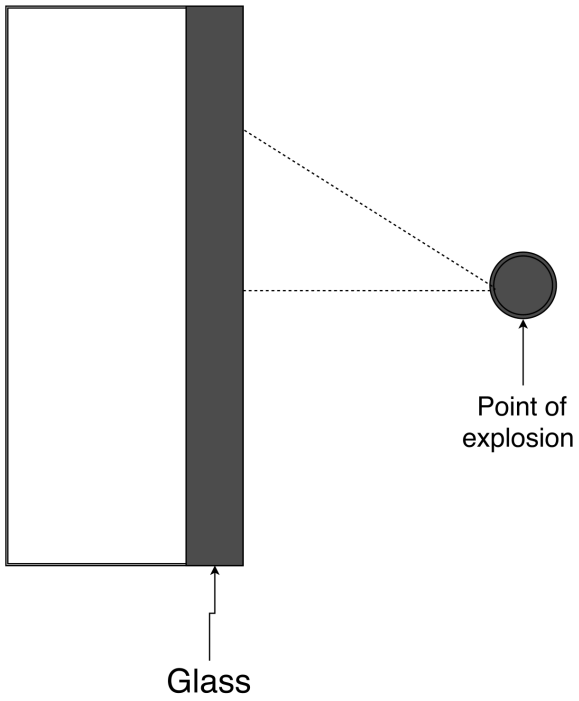
$$\sigma_{ij} = D_{ijkl}\epsilon_{kl}$$







GlassBR



Given

- dimensions of glass plane
- glass type
- explosion characteristics
- tolerable breakage probability

Predict whether the glass will withstand the explosion

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load ($\hat{q}_{\text{tol}}$), Risk of failure (B), ...
- Assumptions: Load duration factor constant,
- Data definitions: relation for B, ...
- Design decisions:
  - Modularity (input module),
  - Implementation Type (Program),
  - Logging (Yes),
  - Input Structure (Bundled),
  - Constant Structure (Inlined),
  - Constant Rep (Constants),
  - Real Number Rep (Double),
  - ...

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load ($\hat{q}_{\text{tol}}$), Risk of failure (B), ...
- Assumptions: Load duration factor constant,
- Data definitions: relation for B, ...
- Design decisions:
  - Modularity (input module),
  - Implementation Type (Program),
  - Logging (Yes),
  - Input Structure (Bundled),
  - Constant Structure (Inlined),
  - Constant Rep (Constants),
  - Real Number Rep (Double),
  - ...

/glassbr /Website/GlassBR_SRS.html /Website/GlassBR_SRS.css /SRS/bibfile.bib /SRS/Makefile /SRS/GlassBR_SRS.tex /SRS/GlassBR_SRS.pdf /src/python /src/python/README.md /src/python/InputParameters.py /src/python/Calculations.py /src/python/Makefile /src/python/doxConfig ...	... /src/java/GlassBR/Calculations.java /src/java/Makefile /src/java/README.md ... /src/cpp/GlassBR /src/cpp/ReadTable.cpp /src/cpp/InputFormat.hpp /src/cpp/Calculations.cpp ... /src/swift/Calculations.swift ... /src/csharp/Control.cs ...

/glassbr
/Website/GlassBR_SRS.html
/Website/GlassBR_SRS.css
/SRS/bibfile.bib
/SRS/Makefile
/SRS/GlassBR_SRS.tex
/SRS/GlassBR_SRS.pdf
/src/python
/src/python/README.md
/src/python/InputParameters.py
/src/python/Calculations.py
/src/python/Makefile
/src/python/doxConfig
...

...
/src/java/GlassBR/Calculations.java
/src/java/Makefile
/src/java/README.md
...
/src/cpp/GlassBR
/src/cpp/ReadTable.cpp
/src/cpp/InputFormat.hpp
/src/cpp/Calculations.cpp
...
/src/swift/Calculations.swift
...
/src/csharp/Control.cs
...

tex

Software Requirements Specification for GlassBR
Nikitha K and Spencer S

Table of Symbols
$\hat{q}_{\text{tol}}$
B
...

Introduction
... The software, herein called GlassBR, ...

Assumptions
ldfConstant: LDF is constant, depends on assumed value of t_d and m, ...

Data Definitions
$$B = \frac{k}{(ab)^{m-1}} \left(Eh^2\right)^m \text{LDF} e^J$$
...

html

sBR

$$B = \frac{k}{(ab)^{m-1}} \left(Eh^2\right)^m \text{LDF} e^J$$

README.md

GlassBR

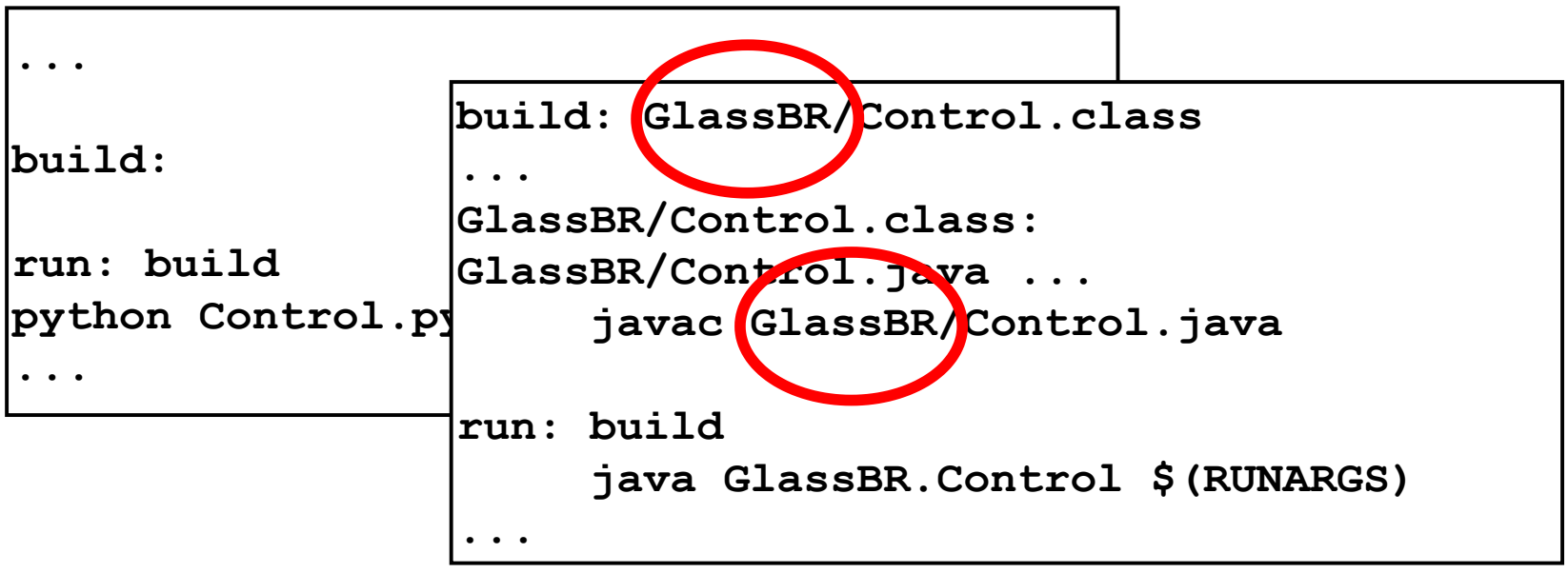
Authors Nikitha K and Spencer S

How to Run the Program: In your terminal command line, enter the same directory as this README file. Then enter the following line

```
make run RUNARGS=input.txt
```

Configuration Files: SDF.txt, TSD.txt must be in the same directory as the executable to run successfully

Versioning: Python Version 3.5.1


...
build:
run: build
python Control.py
...
build: GlassBR/Control.class
...
GlassBR/Control.class:
GlassBR/Control.java ...
javac GlassBR/Control.java
run: build
java GlassBR.Control $(RUNARGS)
...

Calculations.py

```
## \file Calculations.py
# \author Nikitha Krithnan and W. Spencer Smith
# \brie
...
## \bri
# \para
# \para
# \retu
def fun
    out
    pri
    ...
    out

    ret
```

Calculations.java

```
package GlassBR;
/** \file Calculations.java
    \author Nikitha Krithnan and W. Spencer Smith
    \brief Provides functions for calculating the outputs
*/
...
    public static double func_B(InputParameters inParams, double J) throws IOException {
      PrintWriter outfile;
      outfile = new PrintWriter(new FileWriter(new File("log.txt"), true));
      outfile.println("function func_B called with inputs: {");
      ...
      outfile.close();

      return 2.86e-53 /Math.pow(inParams.a * inParams.b, 7.0 - 1.0) *
             Math.pow(7.17e10 * Math.pow(inParams.h, 2.0), 7.0) * inParams.LDF
             * Math.exp(J);
    }
```

J_{tol} in SRS.pdf

Refname	**DD:sdfTol**
Label	Stress distribution factor (Function) based on Pbtol
Symbol	J_{tol}
Units	Unitless
Equation	$$J_{\text{tol}} = \ln\left(\ln\left(\frac{1}{1-P_{\text{btol}}}\right)\frac{\left(\frac{a}{1000}\frac{b}{1000}\right)^{m-1}}{k\left(E\cdot 1000\left(\frac{h}{1000}\right)^2\right)^m LDF}\right)$$
Description	J_{tol} is the stress distribution factor (Function) based on Pbtol (Unitless) P_{btol} is the tolerable probability of breakage (Unitless) a is the plate length (long dimension) (m) b is the plate width (short dimension) (m) m is the surface flaw parameter ($\frac{\text{m}^{12}}{\text{N}^7}$) k is the surface flaw parameter ($\frac{\text{m}^{12}}{\text{N}^7}$) E is the modulus of elasticity of glass (Pa) h is the minimum thickness (m) LDF is the load duration factor (Unitless)

J_{tol} in SRS.tex

```
...
Label & Stress distribution factor (Function) based on
    Pbtol

\\ \midrule \\
Symbol & ${J_{\text{tol}}}$

\\ \midrule \\
Units & Unitless

\\ \midrule \\
Equation & \begin{displaymath}
           {J_{\text{tol}}}=\ln\left(\ln\left(\frac
               {1}{1-{P_{\text{b}\text{tol}}}}\right) \
               frac{\left(\frac{a}{1000} \frac{b}{1000}\
               right)^{m-1}}{k \left(E\cdot{}1000 \left(\
               frac{h}{1000}\right)^{2}\right)^{m} LDF}\
               right)
           \end{displaymath}
\\ \midrule \\
Description & ...
```

J_{tol} in SRS.html

```
...
<th>Equation</th>
<td>
\[{J_{\text{tol}}}=\ln\left(\ln\left(\frac{1}{1-{P_{\text
    {b}\text{tol}}}}\right) \frac{\left(\frac{a}{1000} \
    frac{b}{1000}\right)^{m-1}}{k \left(E\cdot{}1000 \
    left(\frac{h}{1000}\right)^{2}\right)^{m} LDF}\right)
    \]
</td>
...
```

J_{tol} in Python

```
## \brief Calculates stress distribution factor (Function
    ) based on Pbtol
# \param inParams structure holding the input values
# \return stress distribution factor (Function) based on
    Pbtol
def func_J_tol(inParams):
    outfile = open("log.txt", "a")
    print("function func_J_tol called with inputs: {",
        file=outfile)
    print("  inParams = ", end="", file=outfile)
    print("Instance of InputParameters object", file=
        outfile)
    print("  }", file=outfile)
    outfile.close()

    return math.log(math.log(1.0 / (1.0 - inParams.P_btol
        )) * ((inParams.a / 1000.0 * (inParams.b /
        1000.0)) ** (7.0 - 1.0) / (2.86e-53 * (7.17e10 *
        1000.0 * (inParams.h / 1000.0) ** 2.0) ** 7.0 *
        inParams.LDF)))
```

J_{tol} in Java

```
/** \brief Calculates stress distribution factor (
    Function) based on Pbtol
    \param inParams structure holding the input
        values
    \return stress distribution factor (Function)
        based on Pbtol
*/
public static double func_J_tol(InputParameters
    inParams) throws IOException {
    PrintWriter outfile;
    outfile = new PrintWriter(new FileWriter(new File
        ("log.txt"), true));
    ...
    return Math.log(Math.log(1.0 / (1.0 - inParams.
        P_btol)) * (Math.pow(inParams.a / 1000.0 * (
        inParams.b / 1000.0), 7.0 - 1.0) / (2.86e-53
        * Math.pow(7.17e10 * 1000.0 * Math.pow(
        inParams.h / 1000.0, 2.0), 7.0) * inParams.
        LDF)));
}
```

J_{tol} in Drasil (Haskell)

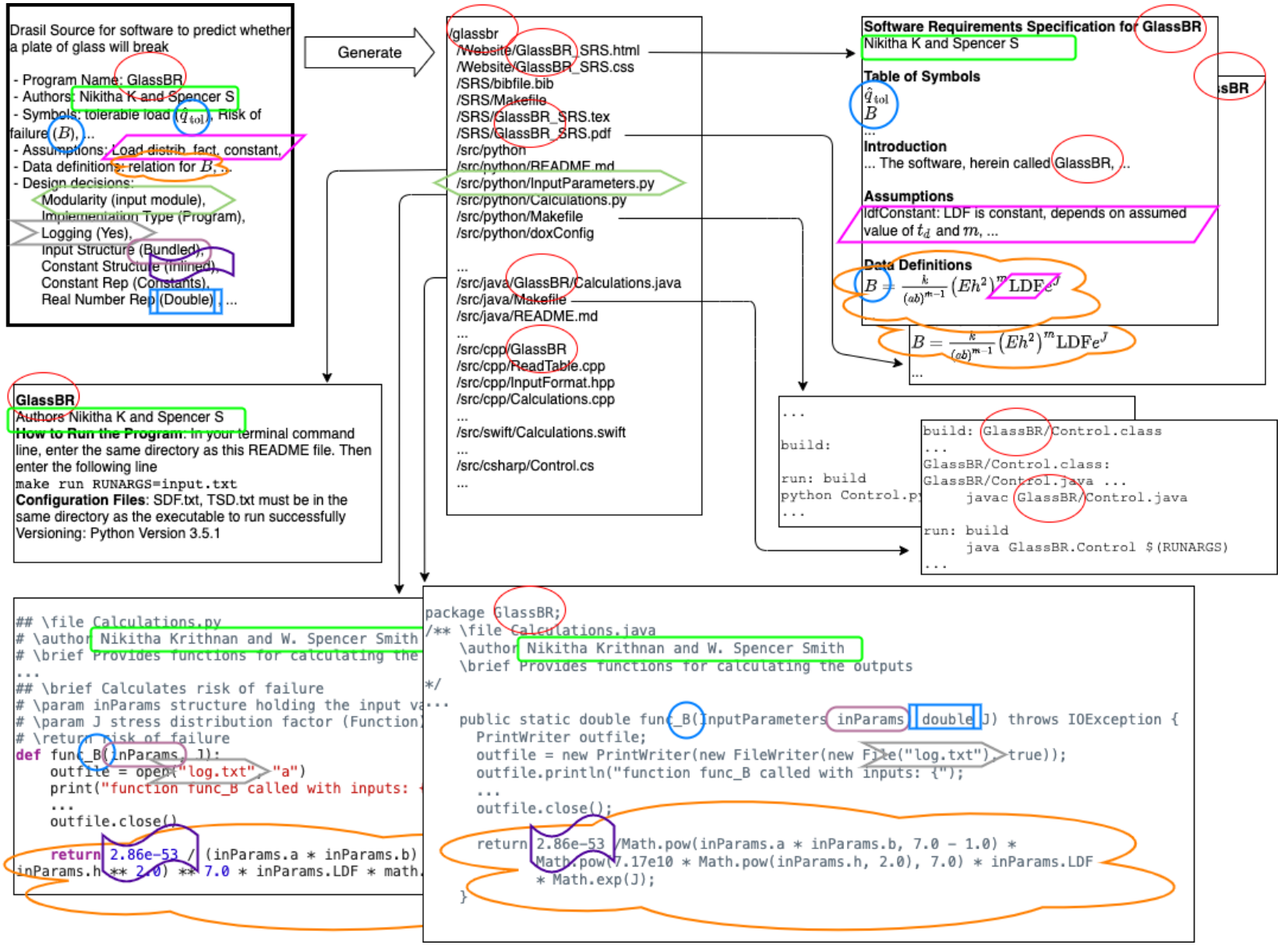
```
tolStrDisFacEq :: Expr
tolStrDisFacEq = ln (ln (recip_ (exactDbl 1 $- sy pbTol))
  `mulRe` (((sy plateLen $/ exactDbl 1000) `mulRe` (sy
     plateWidth $/ exactDbl 1000)) $^ (sy sflawParamM $-
      exactDbl 1) $/
    (sy sflawParamK `mulRe` ((sy modElas `mulRe` exactDbl
        1000 `mulRe`
    square (sy minThick $/ exactDbl 1000)) $^ sy
       sflawParamM) `mulRe` sy lDurFac)))
```

J_{tol} without Unit Conversion

```
tolStrDisFacEq :: Expr
tolStrDisFacEq = ln (ln (recip_ (exactDbl 1 $- sy pbTol))
  `mulRe` ((sy plateLen `mulRe` sy plateWidth) $^ (sy
     sflawParamM $- exactDbl 1) $/
    (sy sflawParamK `mulRe` ((sy modElas `mulRe`
    square (sy minThick)) $^ sy sflawParamM) `mulRe` sy
       lDurFac)))
```

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load ($\hat{q}_{\text{tol}}$), Risk of failure (B), ...
- Assumptions: Load duration factor constant,
- Data definitions: relation for B, ...
- Design decisions:
  - Modularity (input module),
  - Implementation Type (Program),
  - Logging (Yes),
  - Input Structure (Bundled),
  - Constant Structure (Inlined),
  - Constant Rep (Constants),
  - Real Number Rep (Double),
  - ...


Drasil Source for software to predict whether a plate of glass will break
- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load ($\hat{q}_{tol}$), Risk of failure (B), ...
- Assumptions: Load distrib. fact. constant, ...
- Data definitions: relation for B, ...
- Design decisions:
Modularity (input module),
Implementation Type (Program),
Logging (Yes),
Input Structure (Bundled),
Constant Structure (Inlined),
Constant Rep (Constants),
Real Number Rep (Double), ...
Generate
/glassbr
/Website/GlassBR_SRS.html
/Website/GlassBR_SRS.css
/SRS/bibfile.bib
/SRS/Makefile
/SRS/GlassBR_SRS.tex
/SRS/GlassBR_SRS.pdf
/src/python
/src/python/README.md
/src/python/InputParameters.py
/src/python/Calculations.py
/src/python/Makefile
/src/python/doxConfig
...
/src/java/GlassBR/Calculations.java
/src/java/Makefile
/src/java/README.md
...
/src/cpp/GlassBR
/src/cpp/ReadTable.cpp
/src/cpp/InputFormat.hpp
/src/cpp/Calculations.cpp
...
/src/swift/Calculations.swift
...
/src/csharp/Control.cs
...
Software Requirements Specification for GlassBR
Nikitha K and Spencer S
Table of Symbols
$\hat{q}_{tol}$
B
...
Introduction
... The software, herein called GlassBR, ...
Assumptions
ldfConstant: LDF is constant, depends on assumed value of t_d and m, ...
Data Definitions
$B = \frac{k}{(ab)^{m-1}}(Eh^2)^m \text{LDF} e^J$
...
$B = \frac{k}{(ab)^{m-1}}(Eh^2)^m \text{LDF} e^J$
...
GlassBR
Authors Nikitha K and Spencer S
How to Run the Program: In your terminal command line, enter the same directory as this README file. Then enter the following line
make run RUNARGS=input.txt
Configuration Files: SDF.txt, TSD.txt must be in the same directory as the executable to run successfully
Versioning: Python Version 3.5.1
...
build:
run: build
python Control.py
...
build: GlassBR/Control.class
...
GlassBR/Control.class:
GlassBR/Control.java ...
javac GlassBR/Control.java
run: build
java GlassBR.Control $(RUNARGS)
...
\file Calculations.py
\author Nikitha Krithnan and W. Spencer Smith
\brief Provides functions for calculating the
...
\brief Calculates risk of failure
# \param inParams structure holding the input va
# \param J stress distribution factor (Function)
# \return risk of failure
def func_B(inParams, J):
outfile = open("log.txt", "a")
print("function func_B called with inputs: {
...
outfile.close()
return 2.86e-53 / (inParams.a * inParams.b)
inParams.h ** 2.0) ** 7.0 * inParams.LDF * math.
package GlassBR;
/** \file Calculations.java
\author Nikitha Krithnan and W. Spencer Smith
\brief Provides functions for calculating the outputs
*/
...
public static double func_B(InputParameters inParams, double J) throws IOException {
PrintWriter outfile;
outfile = new PrintWriter(new FileWriter(new File("log.txt"), true));
outfile.println("function func_B called with inputs: {");
...
outfile.close();
return 2.86e-53 /Math.pow(inParams.a * inParams.b, 7.0 - 1.0) *
Math.pow(7.17e10 * Math.pow(inParams.h, 2.0), 7.0) * inParams.LDF
* Math.exp(J);
}

Idealized Process

- From wu paper

Capturing Knowledge: Inputs

- Probably won't have time to discuss

Concluding Remarks

- **Take Home Message** — Sustainability and Productivity can potentially be improved via a Generate All Things Approach

References I

Muhammad Usman, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. A product-line model-driven engineering approach for generating feature-based mobile applications. *Journal of Systems and Software*, 123:1–32, 01 2017. doi: 10.1016/j.jss.2016.09.049.

Image Credits

- Apple Product Line Image
- Dodge Lineup