Literate Software and the Drasil Framework

Dan Szymczak

Computing and Software Department
Faculty of Engineering
McMaster University

Thesis Proposal Defense – June 23, 2016

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Literate Software and Drasil

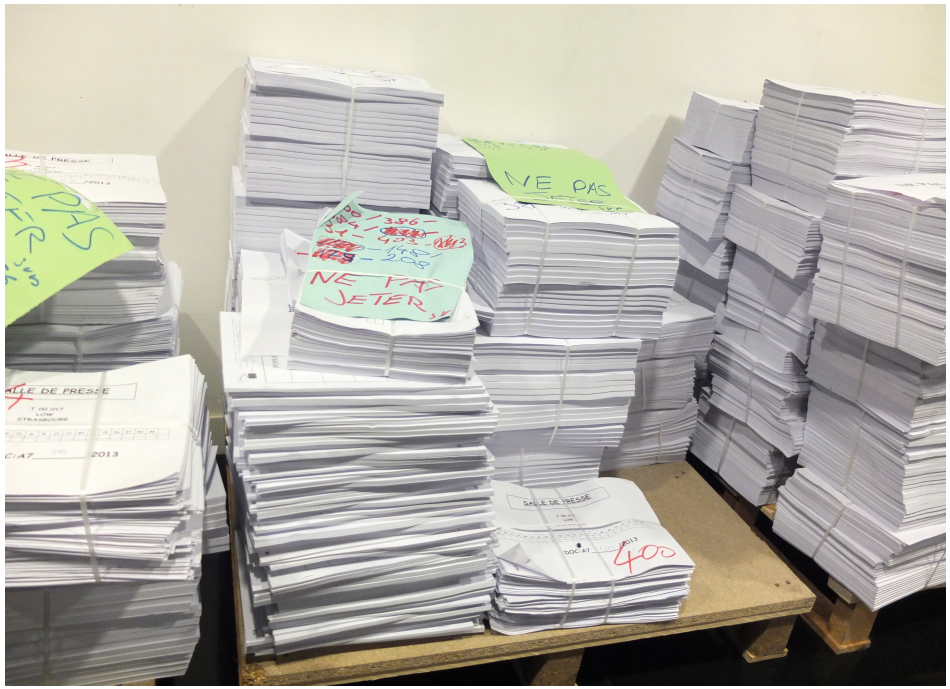
1. Introduction
   - Problem
   - Scope
   - Goals and Objectives
2. State of The Art
3. Current Work
   - Approach
   - The Drasil Framework
4. Next Steps

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Problem

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

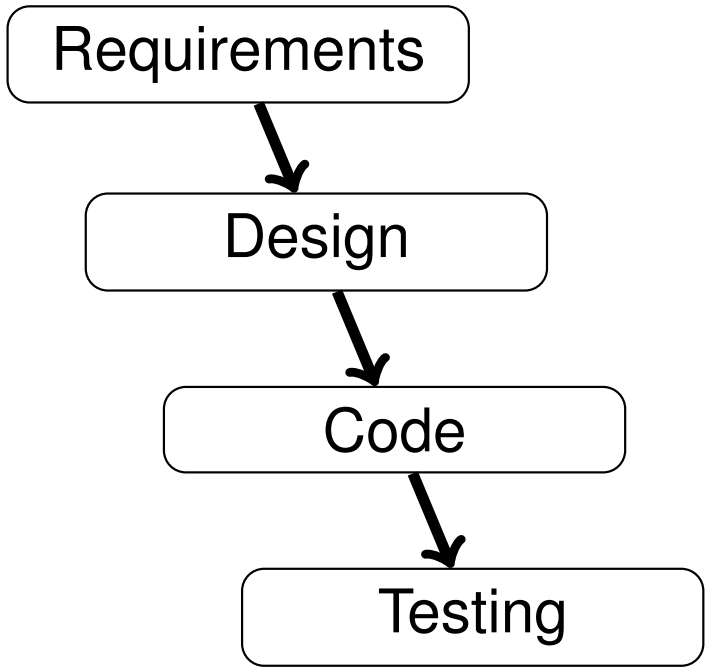
Information Duplication



- Wastes resources
- Reduces software quality

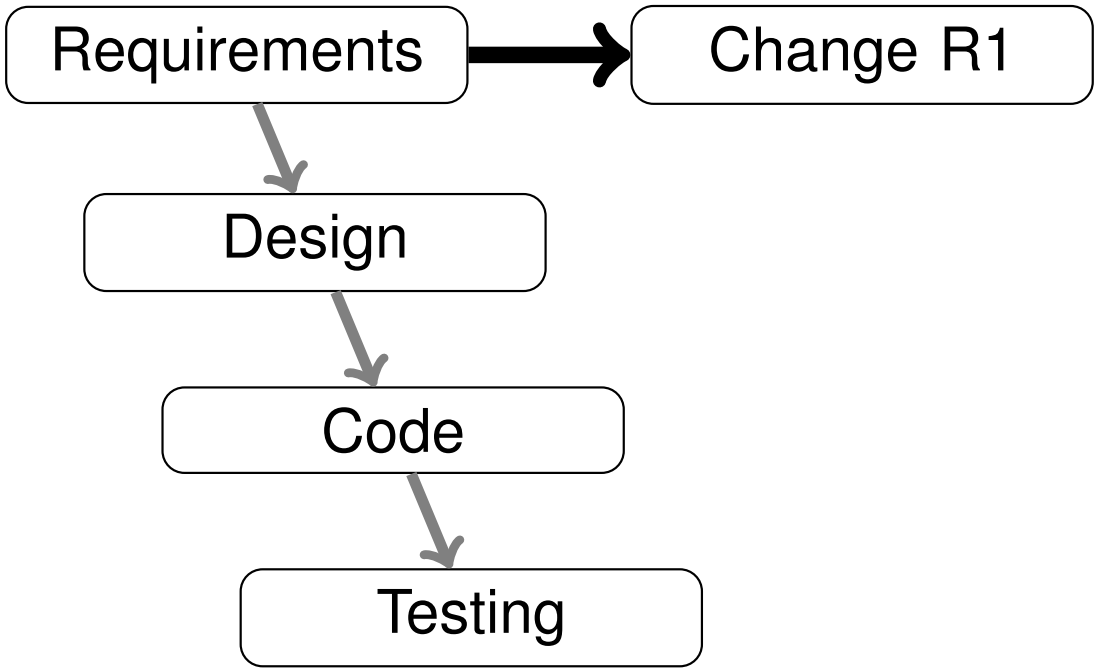
Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Example

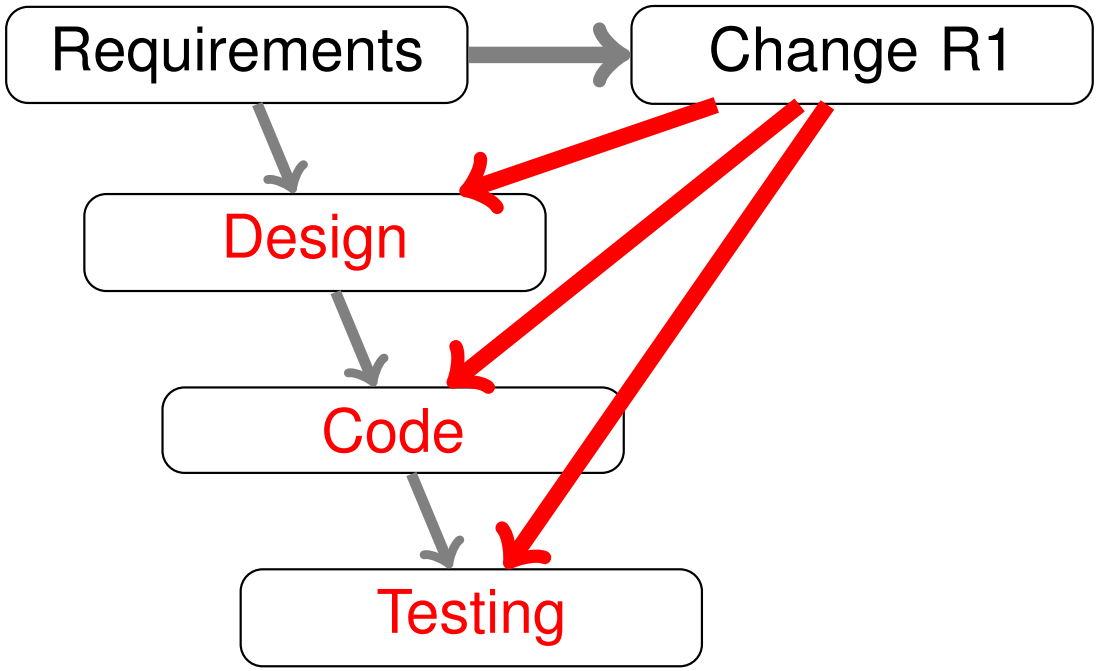


Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Example

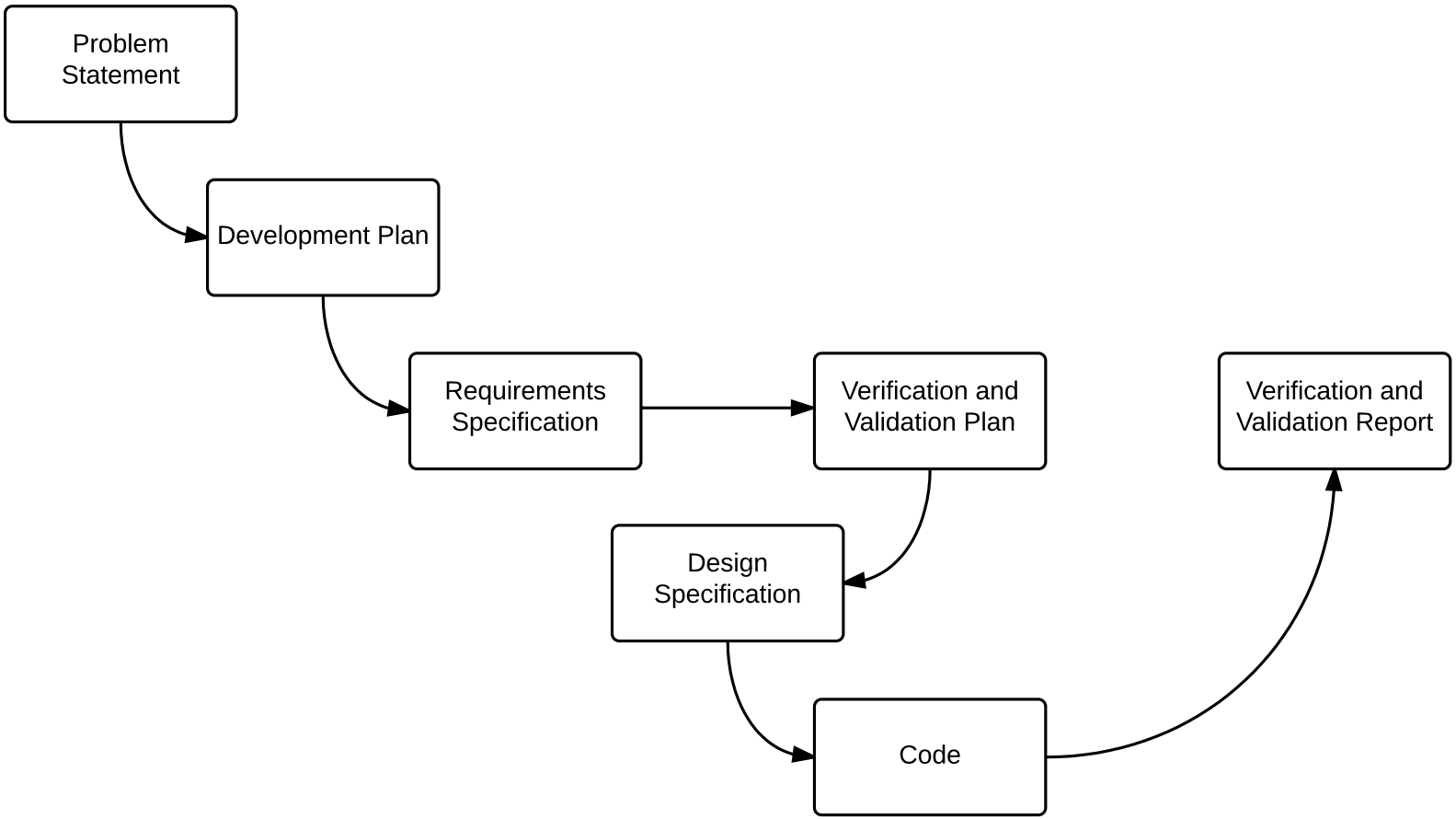


Example



Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Rational Design Process



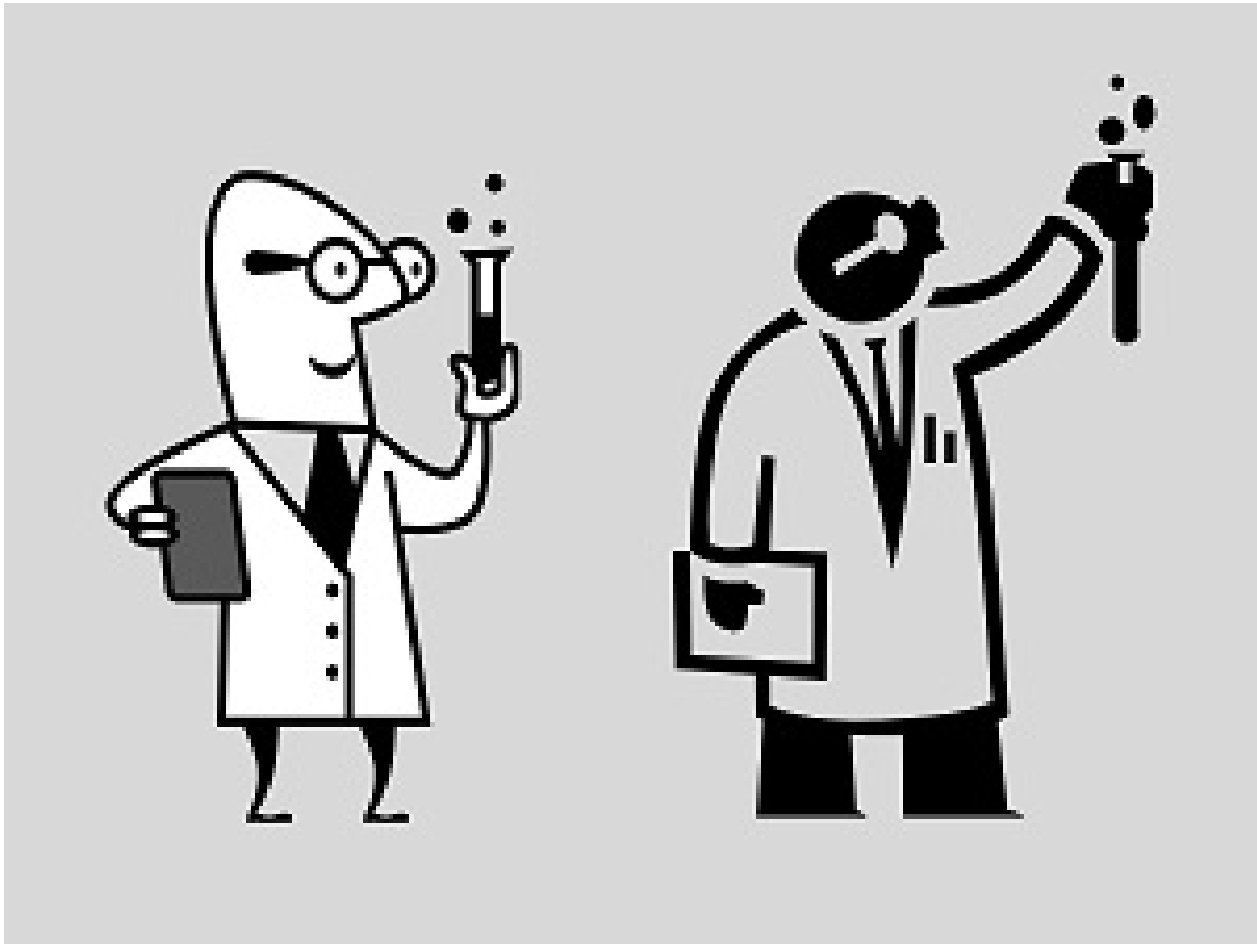
Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Reduced Software Qualities

- Maintainability
- Traceability
- Verifiability
- Reproducibility

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Reproducibility



Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Scope

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Scope

Limited to well-understood domains.

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Scope

Limited to well-understood domains.

Specifically targeting Scientific Computing (SC)

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Scope

Limited to well-understood domains.

Why SC?

- Rich, well-understood background
- SC Developers lack software engineering background

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Goals and Objectives

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Goals and Objectives

Simplify the development process for SC developers

- Create better software artifacts
- Lower the long term cost of updating/maintaining software
- Improve reproducibility

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Goals and Objectives

Create a tool to facilitate a knowledge-based approach

- Create better software artifacts
  - Ensure consistency
- Lower the long term cost of updating/maintaining software
  - Automate creation of software artifacts
- Improve reproducibility

Current State of SC Software Development

- Emphasis on science, not software development [15]
- Rigid, process-heavy approaches are considered unfavourable [4]

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Problems in SC Software Development

- Not enough knowledge reuse
  - Ex. 37 of 52 triangular mesh generators implemented the same triangulation algorithm [26]
- Lack of understanding of software testing [23]
- Very limited tool use (especially version control [42])

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Literate Programming (LP)

- Shifts focus to explain (to humans) what the computer should do [17]
- Algorithms are broken down into *chunks* [12]
- Chunks are ordered to promote understanding
- Many tools for (or inspired by) LP

Advantages of LP

- Increases understandability
- More consistent documentation and code [35]
- Code is more maintainable [29]
- Code can be automatically incorporated into documentation
- Documentation and code are updated simultaneously

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Drawbacks of LP

- Not yet mainstream
- One-source, one document
- Code-centric

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Literate Software Development (LSD)

Combination of LP and Box Structures designed to address:

- Specifying interfaces between modules
- Decomposing boxes
- Implementing designs
- Lack of tool support

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Literate Software Development (LSD)

Combination of LP and Box Structures designed to address:

- Specifying interfaces between modules
- Decomposing boxes
- Implementing designs
- Lack of tool support

LSD's framework *WebBox* addressed the above, but remains primarily code-focused.

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Reproducible Research (RR)

Reproduction can be nearly impossible without the original author's help [11] due to undocumented :

- Assumptions
- Modifications
- Hacks

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Reproducible Research (RR)

Compendia [9] provide a means of encapsulating:

- Research reports
- Data
- Code
- ...

Compendia are intended for use in peer review.

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Approach

Knowledge Based Approach

Knowledge Capture

- Expand chunks
- Specification level encapsulation
- Easy to transform

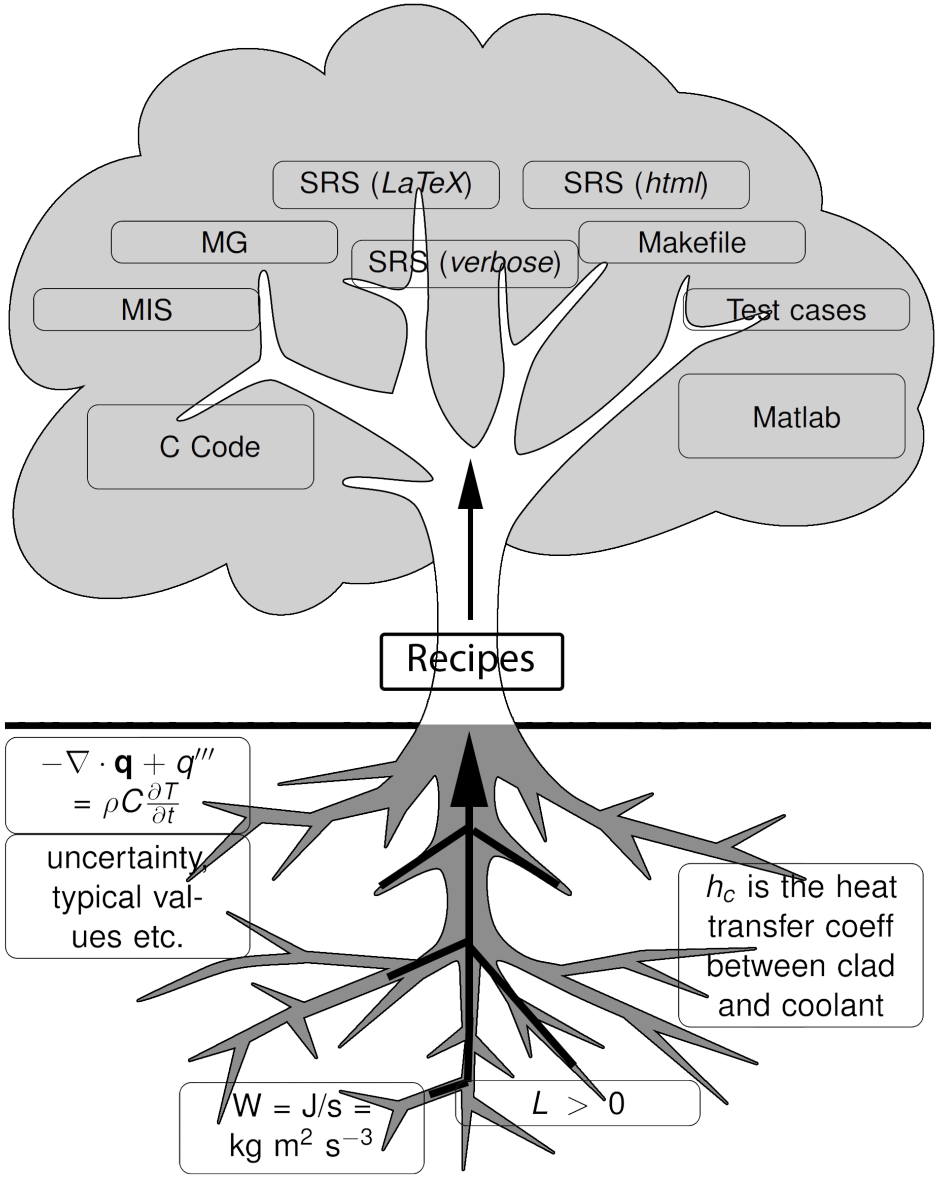
Knowledge Based Approach

Artifact Generation

- Automatic updating
- Traceability & maintainability
- Reproducibility

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Knowledge Based Approach



Knowledge Based Approach

Artifact Generation

- Automatic updating
- Traceability & maintainability
- Reproducibility

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Knowledge Based Approach

Drasil Framework

- Practical, example-driven approach
- Small case studies to start
- Larger case studies in progress

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

The Drasil Framework

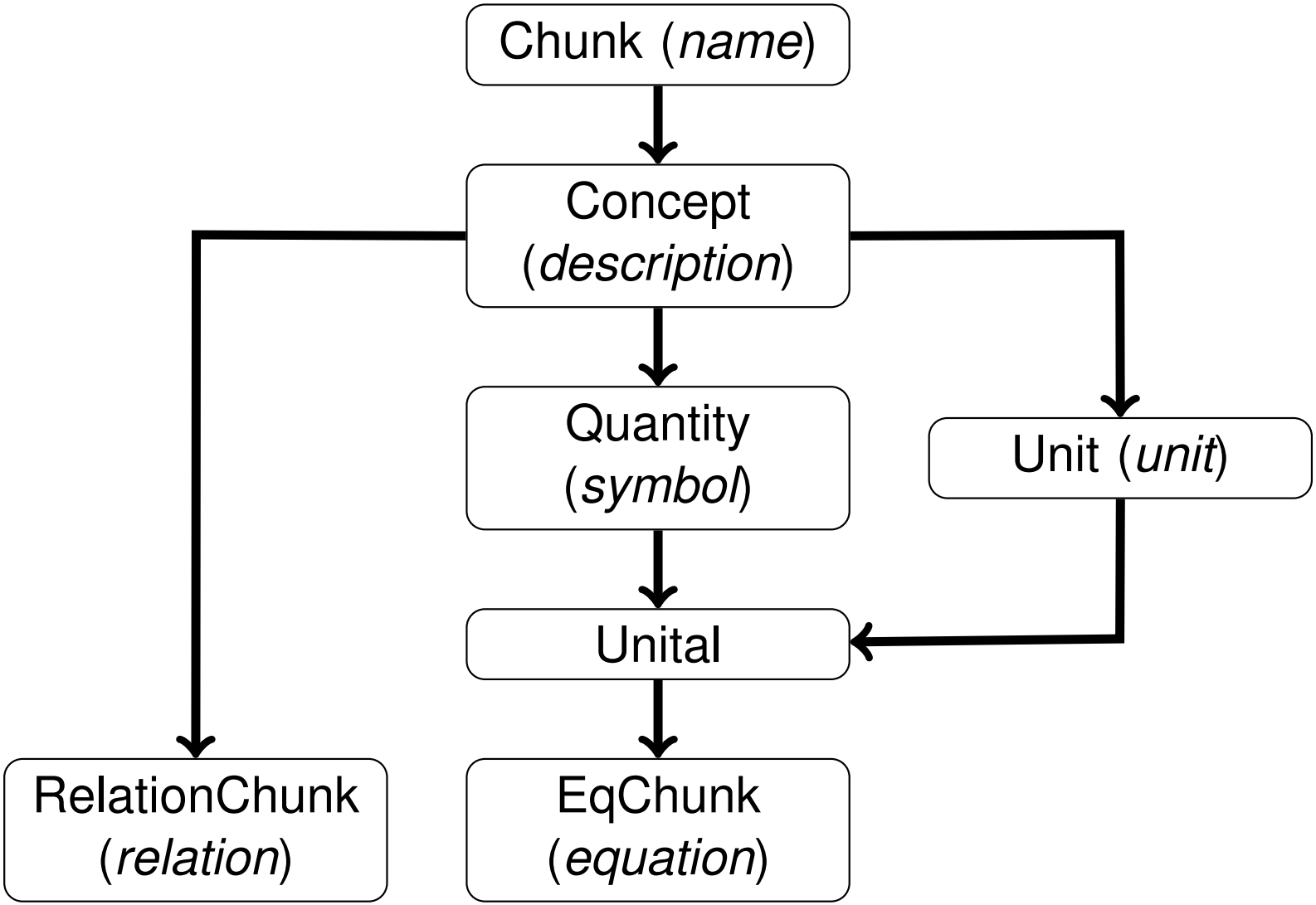
Design

Drasil is currently being implemented as a combination of six eDSLs:

- Expression
- Expression Layout
- Document Layout
- C Representation
- LaTeX Representation
- HTML Representation

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Chunks



Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Example – Fuel Pin SRS

Original SRS from LaTeX
SRS from Generated LaTeX
Generated HTML SRS

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Example – Recipes

```
vars :: [EqChunk]
vars = [h_g, h_c]

s1, s2, s3, s4 :: LayoutObj
s1 = table_of_units si_units
s2 = table_of_symbols vars
s3 = Section 0 (S "Data Definitions") $ map (Definition . Data) vars
s4 = Section 0 (S "Code — Test") $ map (CodeBlock . toCode CLang Calc) [h_c]

srs :: Quantity s => [s] -> String -> [LayoutObj] -> Document
srs ls author body =
  Document ((S "SRS for ") :+:
    (foldr1 (:+:) (intersperse (S " and ") (map (\x -> U $ x ^. symbol) ls))))
    (S author) body

srsBody,lpmBody :: Document
srsBody = srs vars "Spencer Smith" [s1, s2, s3, s4]

lpmBody = Document ((S "Literate Programmer's Manual for ") :+:
  (foldr1 (:+:) (intersperse (S " and ") (map (\x -> U $ x ^. symbol) vars))))
  (S "Spencer Smith") [l1]

l1 :: LayoutObj
l1 = Paragraph (
```

...

Introduction
Problem
Scope
Goals
State of the Art
Current Work
Approach
Drasil
Next Steps
References

Example – Recipes

```
-- Standard code to make a table of units
-- First true example of a (small!) recipe.
module Example.Drasil.Units(table_of_units) where

import Control.Lens ((^.))
import Data.Char (toLower)

import Language.Drasil

table_of_units :: Unit s => [s] -> LayoutObj
table_of_units u = Section 0 (S "Table of Units") [s1_intro, s1_table u]

s1_intro :: LayoutObj
s1_intro = Paragraph
  (S "Throughout this document SI (Syst" :+: (F Grave 'e') :+:
   S "me International d'Unit" :+: (F Acute 'e') :+:
   S "s) is employed as the unit system." :+:
   S " In addition to the basic units, several derived units are" :+:
   S " employed as described below. For each unit, the symbol is" :+:
   S " given followed by a description of the unit with the SI" :+:
   S " name in parentheses.")

s1_table :: Unit s => [s] -> LayoutObj
s1_table u = Table [S "Symbol", S "Description"] (mkTable
  [(\x -> Sy (x ^. unit)),
   (\x -> (x ^. descr) :+: S (" (" ++ map toLower (x ^. name) ++ ")"))
  ] u)
  (S "Table of Units") False
```

Example – Chunks

```
-- {----------- Begin tau_c -----------}

tau_c :: VarChunk
tau_c = makeVC "tau_c" "clad thickness" ((Special Tau_L) `sub` lC)

-- {----------- Begin h_c -----------}

h_c_eq :: Expr
h_c_eq = 2 * (C k_c) * (C h_b) / (2 * (C k_c) + (C tau_c) * (C h_b))

h_c :: EqChunk
h_c = fromEqn "h_c" (S
  "convective heat transfer coefficient between clad and coolant")
  (lH `sub` lC) heat_transfer h_c_eq

-- {----------- Begin h_g -----------}

h_g_eq :: Expr
h_g_eq = (2*(C k_c)*(C h_p)) / (2*(C k_c)+((C tau_c)*(C h_p)))

h_g :: EqChunk
h_g = fromEqn "h_g" (S
  "effective heat transfer coefficient between clad and fuel surface")
  (lH `sub` lG) heat_transfer h_g_eq
```

Example – Common Knowledge

```
metre, second, kelvin, mole, kilogram, ampere, candela :: FundUnit
metre    = fund "Metre"    "length (metre)"                "m"
second   = fund "Second"   "time (second)"                 "s"
kelvin   = fund "Kelvin"   "temperature (kelvin)"          "K"
mole     = fund "Mole"     "amount of substance (mole)"    "mol"
kilogram = fund "Kilogram" "mass (kilogram)"               "kg"
ampere   = fund "Ampere"   "electric current (ampere)"     "A"
candela  = fund "Candela"  "luminous intensity (candela)" "cd"
```

Impressions

Advantages to a knowledge-based approach using Drasil:

1. No inconsistencies inter-/intra-artifact
2. Full traceability
3. Automatic update propagation
4. Reusable knowledge
5. Pervasive bugs

Impressions

Disadvantages:

1. No local hacks
2. Creating common knowledge is difficult
3. Large short-term investment

Next Steps

Planned features:

- More types of information (i.e. physical constraints & reasonable values)
- Generate test cases from constraints
- More output languages (MATLAB)
- More artifact types
- Different document views
- External syntax

References I

[1] Karen S. Ackroyd, Steve H. Kinder, Geoff R. Mant, Mike C. Miller, Christine A. Ramsdale, and Paul C. Stephenson. Scientific software development at a research facility. *IEEE Software*, 25(4):44–51, July/August 2008.

[2] Shereef Abu Al-Maati and Abdul Aziz Boujarwah. Literate software development. *Journal of Computing Sciences in Colleges*, 18(2):278–289, 2002.

[3] Jacques Carette. Gaussian elimination: a case study in efficient genericity with metaocaml. *Science of Computer Programming*, 62(1):3–24, 2006.

References II

[4] Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. Software development environments for scientific and engineering software: A series of case studies. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 550–559, Washington, DC, USA, 2007. IEEE Computer Society.

[5] Michael Deck. Cleanroom and object-oriented software engineering: A unique synergy. In *Proceedings of the Eighth Annual Software Technology Conference, Salt Lake City, USA*, 1996.

[6] Steve M. Easterbrook and Timothy C. Johns. Engineering the software for understanding climate change. *Comuting in Science & Engineering*, 11(6):65–74, November/December 2009.

References III

[7] Matthew Flatt, Eli Barzilay, and Robert Bruce Findler. Scribble: Closing the book on ad hoc documentation tools. In *ACM Sigplan Notices*, volume 44, pages 109–120. ACM, 2009.

[8] Peter Fritzson, Johan Gunnarsson, and Mats Jirstrand. Mathmodelica-an extensible modeling and simulation environment with integrated graphics and literate programming. In *2nd International Modelica Conference, March 18-19, Munich, Germany*, 2002.

[9] Robert Gentleman and Duncan Temple Lang. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, 2012.

[10] Marco S Hyman. Literate c++. *COMP. LANG.*, 7(7):67–82, 1990.

References IV

[11] Cezar Ionescu and Patrik Jansson. Dependently-Typed Programming in Scientific Computing — Examples from Economic Modelling. In *Revised Selected Papers of the 24th International Symposium on Implementation and Application of Functional Languages*, volume 8241 of *Lecture Notes in Computer Science*, pages 140–156. Springer International Publishing, 2012.

[12] Andrew Johnson and Brad Johnson. Literate programming using `noweb`. *Linux Journal*, 42:64–69, October 1997.

[13] Diane Kelly. Industrial scientific software: A set of interviews on software development. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '13, pages 299–310, Riverton, NJ, USA, 2013. IBM Corp.

References V

[14] Diane Kelly. Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *Journal of Systems and Software*, 109:50–61, 2015.

[15] Diane F. Kelly. A software chasm: Software engineering and scientific computing. *IEEE Softw.*, 24(6):120–119, 2007.

[16] Oleg Kiselyov, Kedar N Swadi, and Walid Taha. A methodology for generating verified combinatorial circuits. In *Proceedings of the 4th ACM international conference on Embedded software*, pages 249–258. ACM, 2004.

[17] D. E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

References VI

[18] Jeffrey Kotula. Source code documentation: an engineering deliverable. In *tools*, page 505. IEEE, 2000.

[19] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In *Compstat*, pages 575–580. Springer, 2002.

[20] Russell V Lenth. Statweave users' manual. *URL http://www. stat. uiowa. edu/˜ rlenth/StatWeave*, 2009.

[21] Russell V Lenth, Søren Højsgaard, et al. Sasweave: Literate programming using sas. *Journal of Statistical Software*, 19(8):1–20, 2007.

[22] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

References VII

[23] Zeeya Merali. Computational science: ...error. *Nature*, 467:775–777, 2010.

[24] Harlan D Mills, Richard C Linger, and Alan R Hevner. Principles of information systems analysis and design. 1986.

[25] Nedialko S. Nedialkov. VNODE-LP — a validated solver for initial value problems in ordinary differential equations. Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Ontario, L8S 4K1, 2006.

[26] Steven J. Owen. A survey of unstructured mesh generation technology. In *INTERNATIONAL MESHING ROUNDTABLE*, pages 239–267, 1998.

References VIII

[27] Matthew Patrick, James Elderfield, Richard OJH Stutt, Andrew Rice, and Christopher A Gilligan. Software testing in a scientific research group. 2015.

[28] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[29] Vreda Pieterse, Derrick G. Kourie, and Andrew Boake. A case for contemporary literate programming. In *Proceedings of the 2004 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, SAICSIT '04, pages 2–9, Republic of South Africa, 2004. South African Institute for Computer Scientists and Information Technologists.

References IX

[30] Markus Püschel, José MF Moura, Jeremy R Johnson, David Padua, Manuela M Veloso, Bryan W Singer, Jianxin Xiong, Franz Franchetti, Aca Gačic, Yevgen Voronenko, et al. Spiral: Code generation for dsp transforms. *Proceedings of the IEEE*, 93(2):232–275, 2005.

[31] Norman Ramsey. Literate programming simplified. *IEEE software*, 11(5):97, 1994.

[32] Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico, 1998.

[33] Eric Schulte, Dan Davison, Thomas Dye, Carsten Dominik, et al. A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software*, 46(3):1–24, 2012.

References X

[34] Judith Segal. When software engineers met research scientists: A case study. *Empirical Software Engineering*, 10(4):517–536, October 2005.

[35] Stephen Shum and Curtis Cook. Aops: an abstraction-oriented programming system for literate programming. *Software Engineering Journal*, 8(3):113–120, 1993.

[36] Volker Simonis. Progdoc–a program documentation system. *Lecture Notes in Computer Science*, 2890:9–12, 2001.

[37] Spencer Smith and Nirmitha Koothoor. A document driven method for certifying scientific computing software used in nuclear safety analysis. *Nuclear Engineering and Technology*, Accepted, 2016. 42 pp.

References XI

[38] Spencer Smith, Yue Sun, and Jacques Carette. Comparing psychometrics software development between CRAN and other communities. Technical Report CAS-15-01-SS, McMaster University, January 2015. 43 pp.

[39] Spencer Smith, Yue Sun, and Jacques Carette. Statistical software for psychology: Comparing development practices between CRAN and other communities. *Software Quality Journal*, Submitted December 2015. 33 pp.

[40] W. Spencer Smith, Nirmitha Koothoor, and Nedialko Nedialkov. Document driven certification of computational science and engineering software. In *Proceedings of the First International Workshop on Software Engineering for High Performance Computing in Computational Science and*

References XII

Engineering (SE-HPCCE), page [8 p.], November 2013.

[41] Harold Thimbleby. Experiences of 'literate programming'using cweb (a variant of knuth's web). *The Computer Journal*, 29(3):201–211, 1986.

[42] Gregory V. Wilson. Where's the real bottleneck in scientific computing? Scientists would do well to pick some tools widely used in the software industry. *American Scientist*, 94(1), 2006.