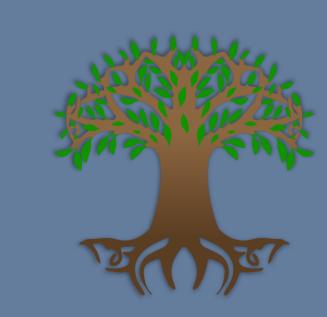


& Software

Zooming Out: The importance of UML diagrams



Noah Cardoso, Dr. Jacques Carette, Dr. Spencer Smith Department of Computing and Software, McMaster University, Hamilton, ON, Canada

Introduction

Goal: To create a visual representation of pieces of knowledge to aid in our understanding of how they are connected in **Drasil**.

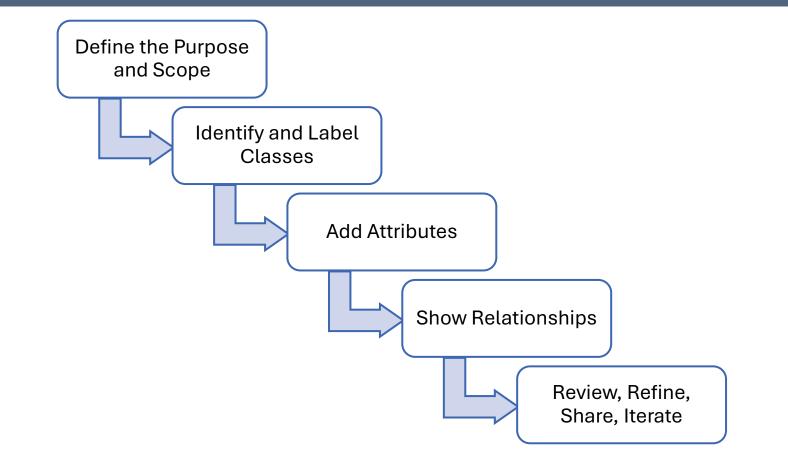
What I did: Created a hierarchy diagram to show how the **Chunks** relate to each other and the information they carry.

Background

What is **Drasil**? Drasil is a software framework written in Haskell that generates all software artifacts (requirements, design, code, tests, build scripts, documentation) based on a single specification in a domain-specific language (DSL).^[1]

What is a **Chunk**? A Chunk is a fragment of knowledge that stores information. Chunks serve as building blocks in a hierarchical system, where each Chunk can be progressively "wrapped" with additional information to create more complex structures.

Design Process



How is it helpful?

Use: It allowed us to see the bigger picture. Made us re-think many of our assumptions about how the Chunks are related to each other.

Benefits:

- Provides a high-level overview of a system's design.
- Helps communicate the structure of the software to developers and stakeholders.
- Serves as documentation for the system's architecture and design.

Diagram ConceptChunk ConstrainedChunk Stage -> Symbol Expresion :: e Explanation :: Sentence Unit Definition: Variable or Function Constraints: Relation :: ModelExpr [ConstraintE] Reference Address :: :: VarOrFunc UnitDefn Concept Domain :: SuitableExprs:: Reasonable Value Maybe UnitDefn Maybe Expr Concept Domain :: UID ShortName UncertainChunk UnitalChunk CodeVarChunk ConstrConcept Pass Value or Reference :: PassBy Constraints : Def Expression : Unit Definition Uncertainty Maybe CodeChunk CodeExpr UnitDefn ConstraintE] any type) Concept Domain Maybe Expr Legend Blue field - what the parent class nests ■ UncertQ ConstrConcep Pink class title - 'drasil-theory' chunk ScopeType Decrement Uncertainty Green class title - `drasil-code` chunk Maybe Derivation hortName White class title - `drasil-lang` chunk \triangleleft deference Address :: String

Figure 1: UML class diagram illustrating the Chunk hierarchy

What are the results?

hanges:

- Replaced ConceptInstance: Introduced an Abbreviations chunk instead.
- Renaming:
- IdeaDict -> Idea
- ConceptChunk -> IdeaDef
- Inheritance and Type System:
- Removed QuantityDict, moved all inheritance to DefinedQuantityDict, and renamed it to Variable.
- Added a type variable for the type system used.
- Simplification:
 - Removed Stage -> Symbol from DefinedQuantityDict/Variable.
 - Removed Maybe UnitDefn from Variable.
- Removed UnitalChunk completely.

Conclusion and Future Work

- The diagram helped to uncover issues within our project which sparked new ideas for a re-design of our current Chunk hierarchy.
- The diagram aided in an issue of the naming convention of our Chunks.
- Figure 2 shows one of the potential redesigns.

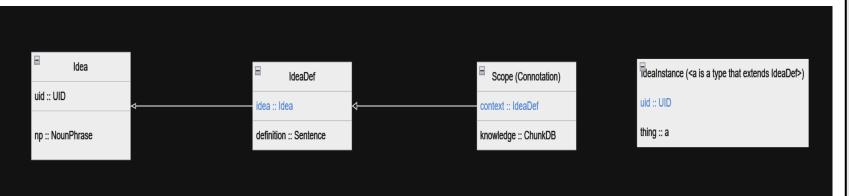


Figure 2: UML class diagram illustrating possible redesign of Chunk hierarchy

References

[1] D. Szymczak, S. Smith and J. Carette, "POSITION PAPER: A Knowledge-Based Approach to Scientific Software Development," 2016 IEEE/ACM International Workshop on Software Engineering for Science (SE4Science), Austin, TX, USA, 2016, pp. 23-26

Acknowledgements

Acknowledgements: I want to acknowledge the support of Dr. Carette and Dr. Smith in making this opportunity available to me and being available for guidance throughout the summer. I would like to acknowledge Jason Balaci for his guidance, for taking the time to help get me up to speed and for providing day—to—day advice and assistance. Lastly, I would like to thank the Dean of Engineering for a portion of my funding this summer