Addressing Knowledge Database Scalability Issues for Software Generation

Drasil – Generate All the Things!

Jack Wyand, Dr. Jacques Carette, Dr. Spencer Smith, Jason Balaci
Department of Computing and Software, McMaster University, Hamilton, ON, Canada
Email: wyandj@mcmaster.ca
LinkedIn: linkedin.com/in/jack-wyand

McMaster University
ENGINEERING
Computing & Software

Background

What is Drasil?

- A framework for generating scientific computing software (SCS) from a stable knowledge base[1]
- Captures and reuses domain knowledge as structured, modular "chunks" to improve traceability, understandability, and to prevent duplication[1]
- Developed mainly in Haskell, with multiple embedded domain-specific-languages (DSLs)[1]

Purpose

- Improve the scalability of Drasil's Chunk Database through deduplication of chunk UIDs within and across chunk tables and the implementation of a new database that does not restrict chunk types

```haskell
-- | Our chunk databases. \Must contain all maps needed in an example.\
-- In turn, these maps must contain every chunk definition or concept
-- used in its respective example, else an error is thrown.
data ChunkDB = CDB {
  -- CHUNKS
    symbolTable           :: SymbolMap
  , termTable             :: TermMap
  , conceptChunkTable     :: ConceptMap
  , _unitTable            :: UnitMap
  , _dataDefnTable        :: DatadefnMap
  , _insmodelTable        :: InsModelMap
  , _gendefTable          :: GendefMap
  , _theoryModelTable     :: TheoryModelMap
  , _conceptinsTable      :: ConceptInstanceMap
  , _citationTable        :: CitationMap
  -- NOT CHUNKS
  , _labelledcontentTable :: LabelledContentMap
  , _refTable             :: ReferenceMap
  , _traceTable           :: TraceMap
  , _refbyTable           :: RefbyMap
  }
makeLenses ''ChunkDB
```

***Figure 1**: Previous implementation of the Chunk Database, where every chunk type needed its own map*

```haskell
{-# LANGUAGE ExistentialQuantification #-}
...
data Chunk = forall a. Typeable a => Chunk a
...
retrieveDQD :: UID -> ChunkDB -> Maybe DefinedQuantityDict
retrieveDQD u cdb = do
    (Chunk expectedDqd) <- lookup u cdb
    cast expectedDqd
```

***Figure 2**: Refined implementation, masking individual chunk types allowing for a single map[2]*

Issues

Chunk UID duplicates

- Duplicates occurred both within and across chunk tables. Across-table duplicates directly block the new implementation because of the following:

Old implementation

```
acceleration :: QuantityDict
acceleration :: ConceptChunk
```

New implementation

ChunkDB	
UID	Type
"acceleration"	QuantityDict
"acceleration"	ConceptChunk

***Figure 3**: Searching for the ConceptChunk with the UID "acceleration" will return Nothing since the QuantityDict appears first*

Methods

- Fixing UID duplicates – within tables

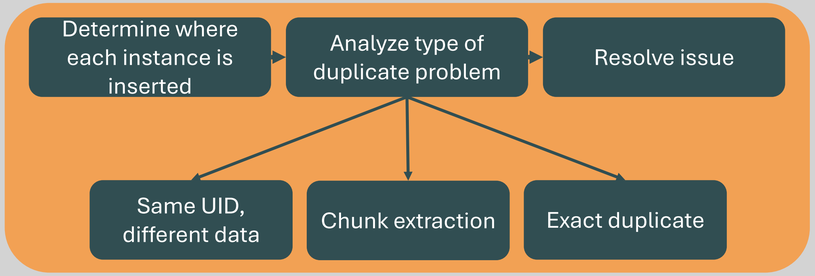

***Figure 4**: The general method of fixing within-table duplicates*

- Fixing UID duplicates – across tables

Constructor	Type	Constructs	File	Line	Permalink	Fixed
qw	Extractor	QuantityDict	drasil-lang\lib\Language\	61	https://github.co	☑
mkQuant	Constructor	QuantityDict	drasil-lang\lib\Language\	66	https://github.co	☑
mkQuant'	Constructor	QuantityDict	drasil-lang\lib\Language\	72	https://github.co	☑
implVar	Constructor	QuantityDict	drasil-lang\lib\Language\	77	https://github.co	☑
implVar'	Constructor	QuantityDict	drasil-lang\lib\Language\	85	https://github.co	☑
implVarUID	Constructor	QuantityDict	drasil-lang\lib\Language\	93	https://github.co	☑
implVarUID	Constructor	QuantityDict	drasil-lang\lib\Language\	101	https://github.co	☑
vc	Constructor	QuantityDict	drasil-lang\lib\Language\	109	https://github.co	☑
vcUnit	Constructor	QuantityDict	drasil-lang\lib\Language\	113	https://github.co	☑
vcSt	Constructor	QuantityDict	drasil-lang\lib\Language\	117	https://github.co	☑
vc''	Constructor	QuantityDict	drasil-lang\lib\Language\	133	https://github.co	☑
tm	Constructor	TheoryModel	drasil-theory\lib\Theory\	114	https://github.co	☑
tmNoRefs	Constructor	TheoryModel	drasil-theory\lib\Theory\	124	https://github.co	☑
qwUC	Constructor	Input	drasil-theory/lib/Theory/	112	https://github.co	☑
qwC	Constructor	Input	drasil-theory/lib/Theory/	116	https://github.co	☑
mkUnitary	Constructor	UnitaryChunk	drasil-lang\lib\Language\	63	https://github.co	☑

***Figure 5**: Tracking the use cases of all the QuantityDict constructors*

Results

- **1425** UID duplicates fixed overall
- QuantityDict type replaced by DefinedQuantityDict, eliminating all symbol-concept extraction duplicates
- Work revealed many problems and insights within Drasil, significantly contributing to future improvements

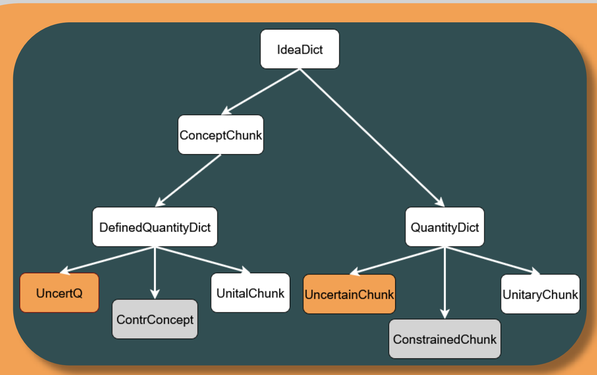

***Figure 6**: The previous chunk structure*

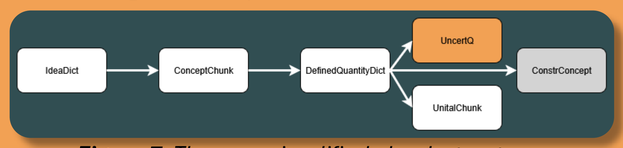

***Figure 7**: The new, simplified chunk structure*

Conclusions

With the improved chunk database implementation, the scalability of future case studies generated with Drasil has been vastly improved. Furthermore, the overall quality of Drasil has been setup to be greatly enhanced through the many issues that my work has uncovered.

References

[1] D. Szymczak, S. Smith, and J. Carette, 'Position paper: A knowledge-based approach to scientific software development', in *Proceedings of the International Workshop on Software Engineering for Science*, Austin, Texas, 2016, pp. 23–26.

[2] J. Balaci, "Adding types and theory kinds to Drasil", M.Sc. Thesis, Dept. Computing and Software, McMaster Univ., Hamilton, ON, Canada, 2022. [Online] . Available: http://hdl.handle.net/11375/29574

View the project here!

https://github.com/JacquesCarette/Drasil