Data Format Description Language &

APACHE Daffodil™ (incubating)

Killing the Data Format Problem Forever Using Apache Spark with DFDL

Mike Beckerle, Data Archeologist, Tresys Technology
mbeckerle at tresys.com or at apache.org

TRESYS Deep.

Goal For Today

My goal in this talk is to convince you that....

- DFDL/Daffodil solves the data format problem
- It is easy to use with Spark

And...

- Daffodil is technology that would be *fun* to work on
  - *Build Community* for Apache Daffodil (Incubating)
  - Recruit developers, esp. Scala developers ♥

APACHE Daffodil™

TRESYS Deep.

Agenda

Motivation

- The Data Format Problem
- DFDL, standards, Daffodil, and why it is important
- DFDL Schemas

Technology

- DFDL (pronounced “DaFoDiL” or “Dee Eff Dee Ell”)
- Daffodil – the software
  - Code base details and status

Demonstration

- Daffodil with Spark

Conclusion

- Call for contributors and users

Why is DFDL Needed?

There are *hundreds* of ad-hoc data format description systems

Every Enterprise Software Company

- IBM (10+)
- Oracle(10+)
- SAP(10+)
- Microsoft
- SAS
- SyncSort
- AbInitio
- Pervasive
- Qlik/Expressor

…. Dozens more

Every kind of software that takes in data:

- data directed routing
- database
- data analysis and/or data mining
- data cleansing
- master data management
- application integration

All these data format descriptions are:

- ***proprietary***
- ***ad-hoc***
- ***incompatible***

Even within products of the same company!

Why DFDL is Needed?

Hundreds of data format description systems… means:

- Investment is spread too thin
  - Tools for creating data formats are inadequate
  - No product is comprehensive enough
    - Difficulty is grossly underestimated
  - Some products aren't fast enough
- Customer lock in
- Inflexible packaging
  - Not libraries - must embed some product in your application data flow

Why DFDL is Needed - New Use Cases

Cybersecurity

- Normalization of data
  - Complete rip and rebuild
  - Break down data fully based on DFDL schema
  - Validate at fine granularity
  - Reassemble according to DFDL schema
- Removes a large class of data-borne threats
  - Data adheres to format spec. exactly!
  - Reduces the "attack surface" for software processing it.

Data Publishing

- Open Data mandates

Why DFDL is Needed - New Use Cases

Modernization

- Legacy data systems are still the source, *and target* for much processing
- Coexistence is required for successful incremental modernization

Skills Leverage

- Developers with XML/JSON skills are available
- Legacy data format skills are precious
  - Military data formats - Link16, VMF, USMTF, ...
  - COBOL and other FINSERV formats

Solving the Data Format Problem

An Open Standard for DFDL

- Multiple implementations that interoperate
  - Commercial & Open Source
- Long-term sponsors
  - IBM – has their own DFDL implementations
  - US DoD, Canada DND
    - Cybersecurity
- Available DFDL schemas for important data formats

A High-Quality Open Source Library Implementation

- With a supporting community of developers
- With available commercial support (Tresys)

APACHE Daffodil™

TRESYS Deep.

Data Format Description Language

DFDL: A new open standard

- From the Open Grid Forum (OGF)
- Work began in **2001**, accelerated around 2008
- Major contributors from UK, Canada, and USA
- First Implementation: IBM - November 2011
    - Business-oriented subset of DFDL language
- DFDL Specification - Version 1.0 – Sept. 2014
    - Thick - about 200 pages if you print it.
    - Allows "conforming subsets" - required core is small
- Proposed Recommendation - Status (as of Oct 2016)
    - Waiting for two-implementation interoperability demonstration
- DFDL Workgroup is active
    - Clarifications, Errata on v1.0

Data Format Description Language

DFDL is a way of *describing* data formats

- It is NOT a data format itself!

DFDL combines State-of-the-Art

- Union of capabilities across many marketplace data integration products/tools/packages

DFDL adds small number of real innovations

- Overcome limitations of prior-gen e.g.,
  - Computed Elements Capability
  - Expression language
  - BitOrder

Data Format Description Language

Core Concepts

- Leverage XML Schema (XSDL)
  - Grammar scaffolding
  - Describes the *logical* data model
  - DFDL uses only a *subset* of XML schema
  - Provides standard ways to annotate
- Add annotations
  - Describe the *physical* representation.
- Read and write from same DFDL Schema

Because Developers [Love | Hate] XML

- The DFDL Schema is based on XSDL
- The Infoset created when parsing data does NOT have to be XML

Things DFDL (v1.0) Does

DFDL is for <u>Data Sets</u>

- Things typically thought of as files full of data about XYZ.
  - Rows, Tables
  - Header-Body-Trailer
  - Hierarchical / Nested record-oriented data
  - Messages
- Industry & Military data interchange formats
  - HL7, HIPAA-5010, SWIFT, NACHA, X25, Link16, etc.

Things DFDL (v1.0) Doesn't Do

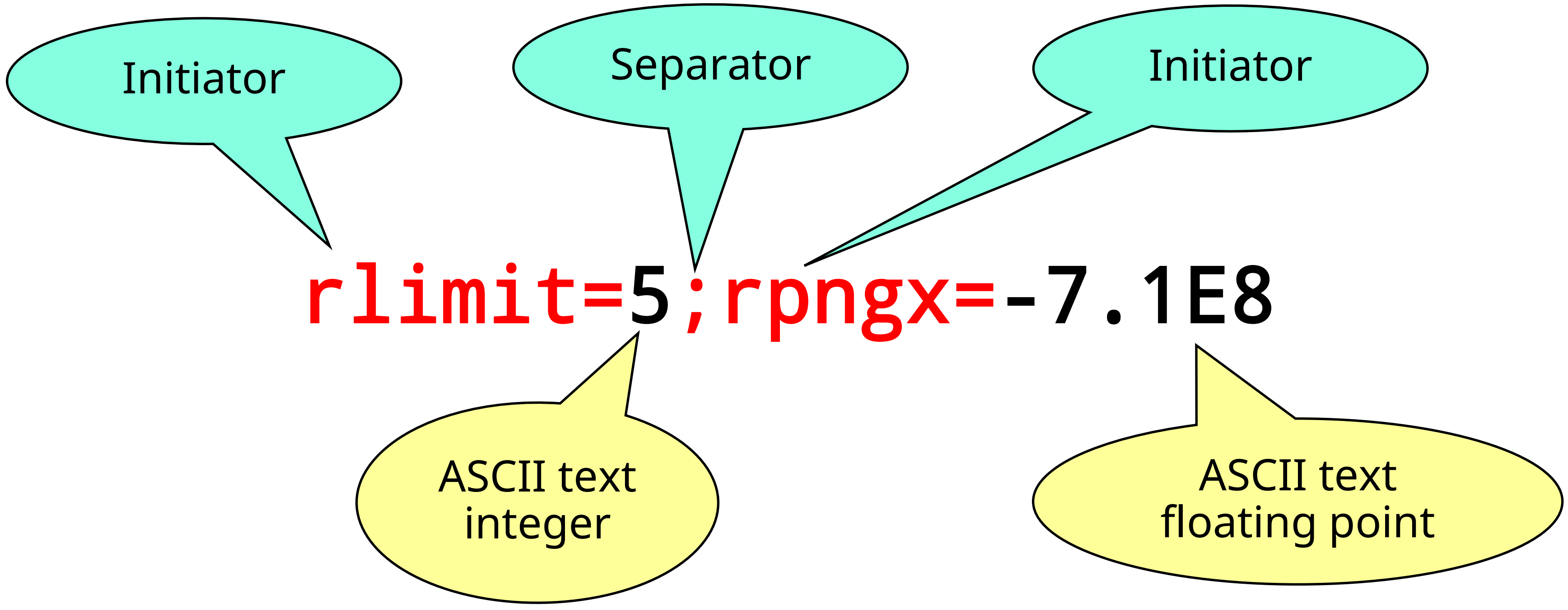
- DFDL v1.0 was ***not*** *originally* intended for:
  - Document file formats

    e.g., MS Office documents (.doc), or RTF, PDF
  - Archives like zip files or tar files
  - Core dump/memory image format
  - Storage format of a RDBMS table
  - Graphs of pointers - object graphs dumped from memory

Example – Delimited Text Data

```
rlimit=5;rpngx=-7.1E8
```

Example – Delimited Text Data



Separators, initiators (aka tags), & terminators are all examples in DFDL of *delimiters*

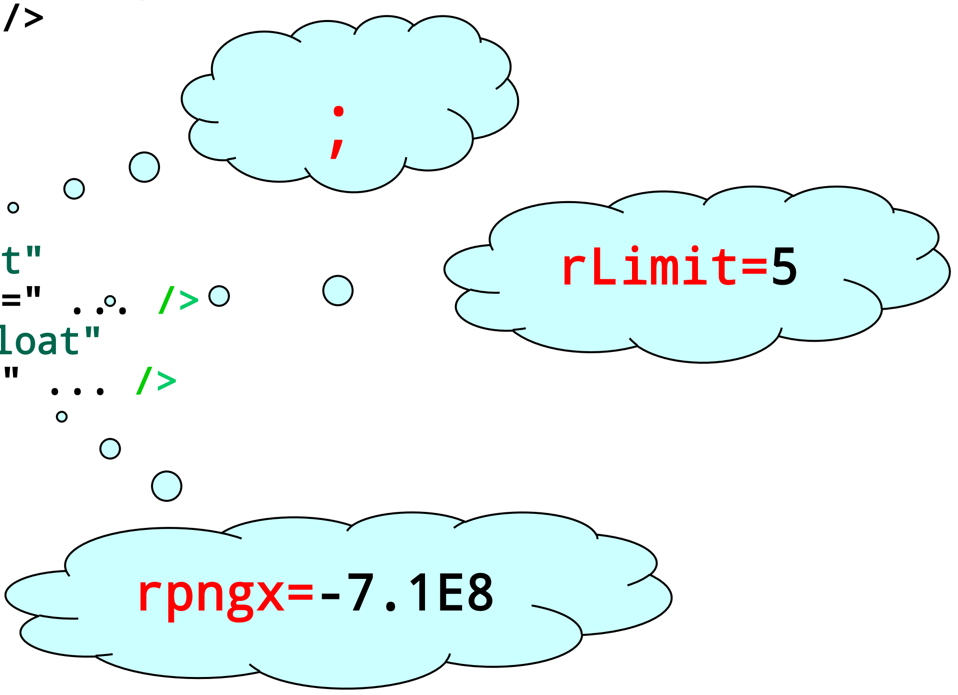
DFDL Schema

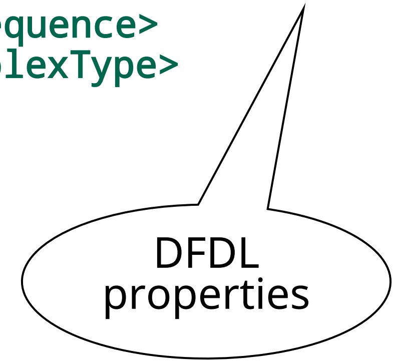
```
<xs:complexType name=“rValues">
  <xs:sequence>
    <xs:element name=“rlim" type="xs:int"/>

     <xs:element name=“rpng" type="xs:float"/>

  </xs:sequence>
</xs:complexType>
```

Logical Elements

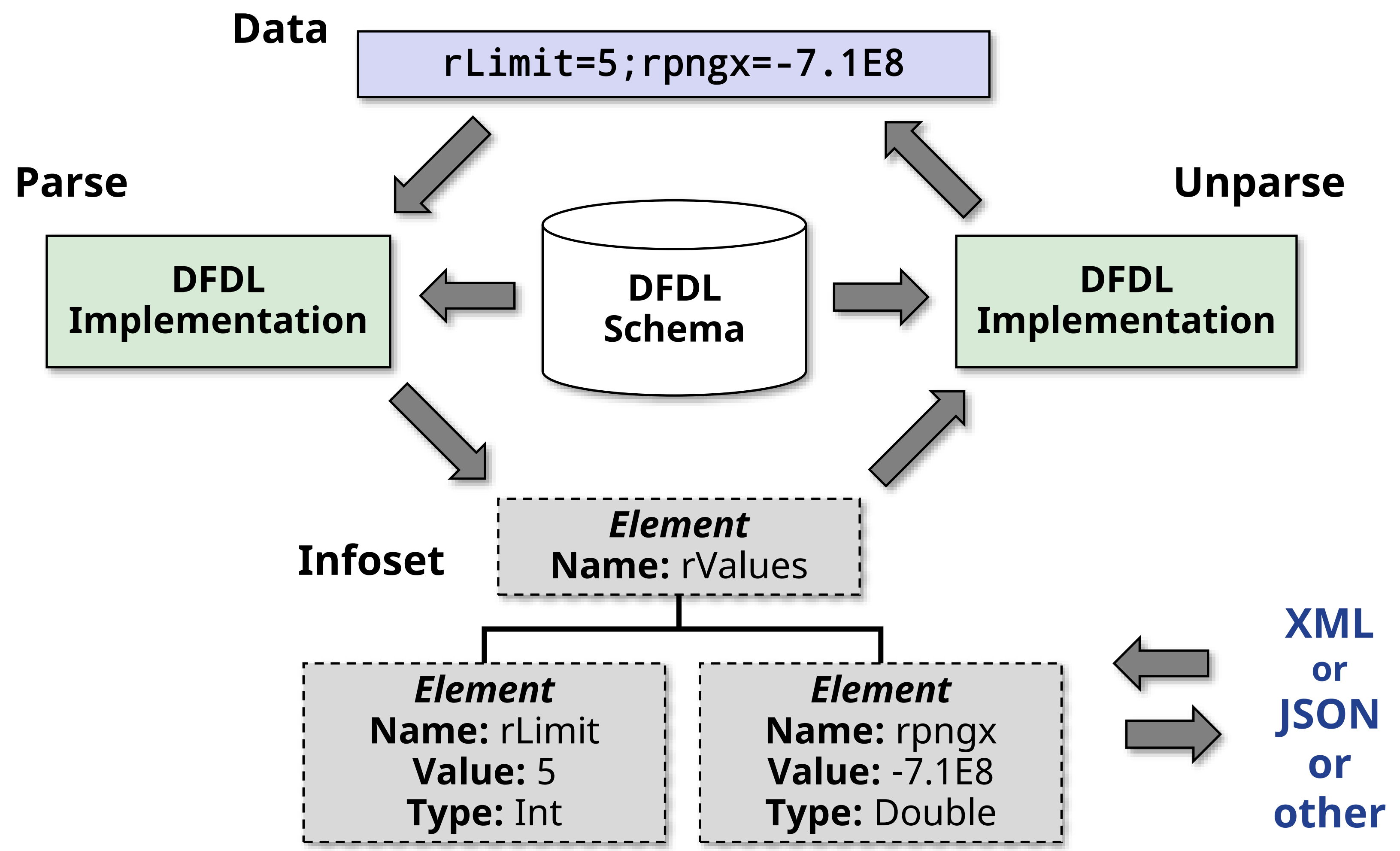
DFDL schema

```
<xs:annotation>
  <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
     <dfdl:format representation="text"
             textNumberRep="standard" encoding="ascii"
             lengthKind="delimited“ .../>
   </xs:appinfo>
</xs:annotation>

<xs:complexType name=“rValues">
  <xs:sequence dfdl:separator=";" ... >
    <xs:element name="rLim" type="xs:int"
                dfdl:initiator=“rLimit=" ... />
     <xs:element name=“rpng" type="xs:float"
                dfdl:initiator=“rpngx=" ... />
  </xs:sequence>
</xs:complexType>
```





DFDL Data and Infoset Lifecycle



More DFDL Properties

initiator
terminator
documentFinalTerminatorCanBeMissing
outputNewLine
length
lengthPattern
textStringPadCharacter
textNumberPadCharacter
textCalendarPadCharacter
textBooleanPadCharacter
escapeCharacter
escapeBlockStart
escapeBlockEnd
escapeEscapeCharacter
extraEscapedCharacters
textNumberPattern
textStandardGroupingSeparator
textStandardDecimalSeparator
textStandardExponentRep
textStandardInfinityRep
textStandardNaNRep
textStandardZeroRep
textBooleanTrueRep
textBooleanFalseRep
calendarPattern
calendarLanguage
binaryCalendarEpoch
nilValue
separator
occursStopValue

inputValueCalc
outputValueCalc

textBidi
textBidiTextOrdering
textBidiOrientation
textBidiSymmetric
textBidiTextShaped
textBidiNumeralShapes

byteOrder
bitOrder
encoding
encodingErrorPolicy
utf16Width
ignoreCase

alignment
alignmentUnits
fillByte
leadingSkip
trailingSkip

lengthKind
lengthUnits
prefixIncludesPrefixLength
prefixLengthType

representation

textPadKind
textTrimKind
textOutputMinLength

escapeKind
generateEscapeBlock

textStringJustification
textNumberRep
textNumberJustification

textNumberCheckPolicy
textStandardBase
textNumberRoundingMode
textNumberRounding
textNumberRoundingIncrement
textZonedSignStyle

binaryNumberRep
binaryDecimalVirtualPoint
binaryNumberCheckPolicy
binaryPackedSignCodes
binaryFloatRep

textBooleanJustification

binaryBooleanTrueRep
binaryBooleanFalseRep

textCalendarJustification

calendarPatternKind
calendarCheckPolicy
calendarTimeZone
calendarObserveDST
calendarFirstDayOfWeek
calendarDaysInFirstWeek
calendarCenturyStart
binaryCalendarRep

nilKind
nilValueDelimiterPolicy

useNilForDefault
emptyValueDelimiterPolicy

sequenceKind
hiddenGroupRef

initiatedContent

separatorPosition
separatorPolicy
separatorSuppressionPolicy

choiceLengthKind
choiceLength
choiceDispatchKey
choiceBranchKey

occursCountKind
occursCount

floating
truncateSpecifiedLengthString

decimalSigned

APACHE Daffodil™ TRESYS Deep.

DFDL Schemas

*   = in development
** = not yet published

Public (github)	MIL-STD-2045 PCAP NITF PNG JPEG NACHA vCard ShapeFile(.shp) QuasiXML GeoNames	EDIFACT IBM4690-TLOG ISO8583 BMP GIF Praat TextGrid ARINC429* JPEG2000** planned: EP, DNG, WMF, EMF, … planned: Asterisk, IPFIX
FOUO (DI2E.net & Forge.mil)	VMF (MIL-STD-6017) USMTF ATO (MIL-STD-6040) LINK16 (NATO STANAG 5516/MIL-STD-6016) A-GNOSC REMEDY ARMY DRRS USCG UCOP CEF-R1965 GMTIF (STANAG 4607)	
Commercial License $$$	SWIFT-MT (IBM) HIPAA-5010 (IBM) HL7-2.7 (IBM)	

APACHE Daffodil™

TRESYS Deep.

Other DFDL Implementations

- IBM DFDL - The First DFDL Implementation - v1.0 Nov 2011
  - Embeddable as Library, C and Java
  - Includes Eclipse based tooling - graphical DFDL schema editor/test environment
  - Found in IBM products:

    IBM App Connect Enterprise product family

    IBM InfoSphere Master Data Management

    IBM z/TPF product family
- European Space Agency - DFDL4S = DFDL for Space
  - Embeddable as Library, Java and C++ versions.
  - Binary-data-only subset of DFDL
  - Created by ESA for satellite data descriptions
  - Evolving to be a fully compatible DFDL subset.
  - More info at
    - http://eop-cfi.esa.int/index.php/applications/dfdl4s

APACHE
Daffodil™

The Open Source DFDL Implementation

aka Apache Daffodil (Incubating)

APACHE Daffodil™

TRESYS Deep.

Daffodil - History

- Started out at University of Illinois/NCSA
  - Research project ~2009
  - Written in **Scala** - runs on Java JVM
- Further developed by Tresys Technology ~2012
  - Funded by the US DoD, Canada DND
  - Open source from the start
  - Version 1.0 – parse only, XML – 2015-03
  - Version 2.0 – parse & unparse, XML + JSON – 2017-09
- Apache Incubator - started 2017-08
  - Version 2.1.0 – 2018-05
  - Version 2.2.0 - available now

Avoid Version Confusion

DFDL Language Specification

- v1.0

Daffodil Software

- v1.0.0
- v1.1.0
- v2.0.0
- v2.1.0
- v2.2.0
- *v2.3.0*
- *v3.x*
- ….

Daffodil

If you download it, what do you get?

- Jar libraries – runs on JVM
  - Compiler, runtime, utilities, TDML runner
- Command Line Interface
  - Interactive CLI debugger and trace
- Java & Scala API with documentation
- XML and JSON for parse-output, unparse-input

You must get your DFDL schemas somewhere...

- github (DFDLSchemas, others)
- Daffodil/DFDL project on DI2E.net/Forge.mil
- Write them! (and share them!)

Daffodil Internal Components

Compiler
- compiles DFDL schemas to runtime data structures
- Issues diagnostics

Scala API

Java API

Command Line Interpreter

Runtime

DPath - Xpath-like language
- compiler
- runtime

Parser primitives

Infoset
- Convert to/from XML, JSON
- Fast, constant-time access

Unparser primitives

Breakpoint debugger

TDML Runner
- Test (& Tutorial) Data Markup Language

Tests - Unit (Scala)

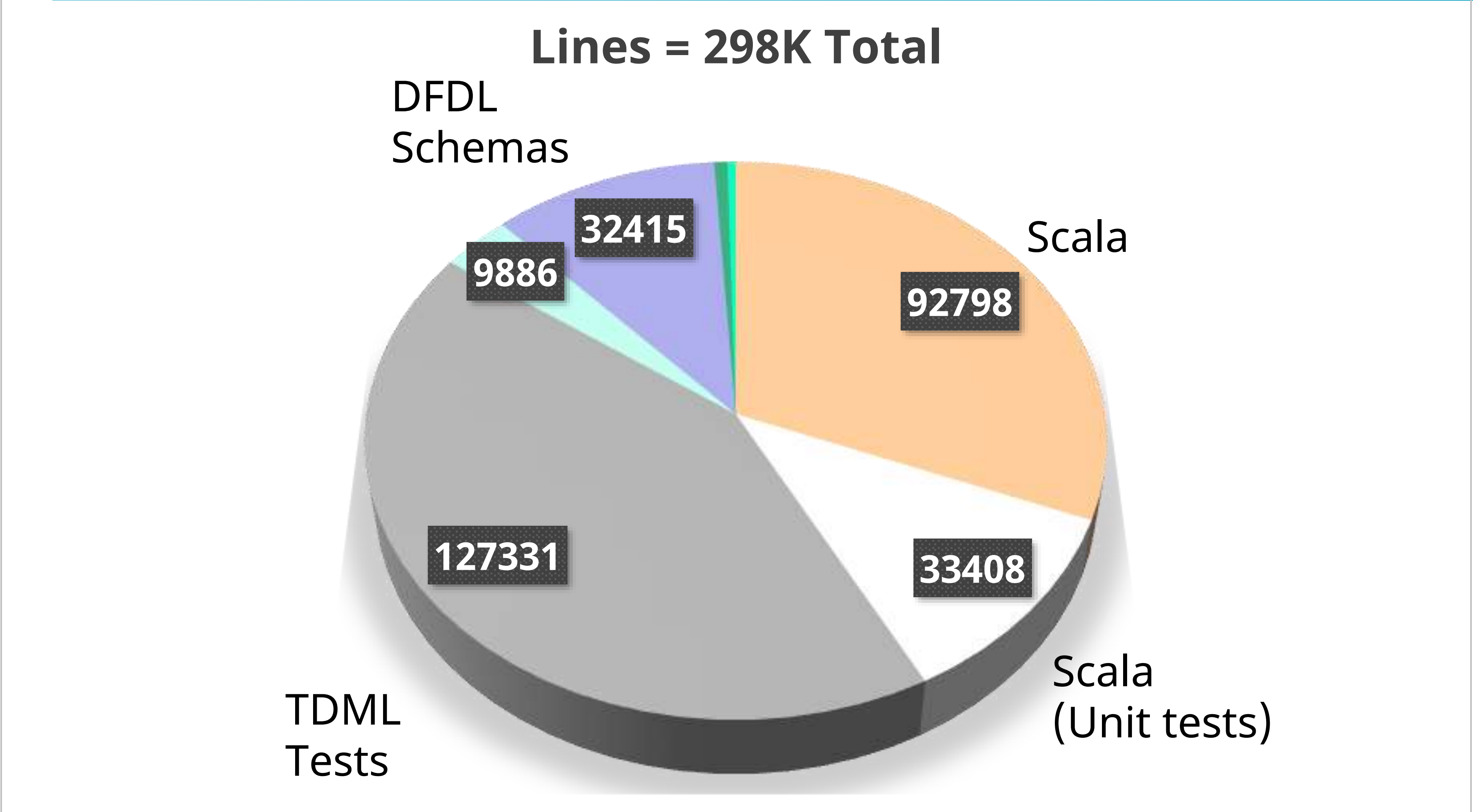
Tests - System (TDML)

Utility Libraries
- for compiler
- for runtime

I/O library
- bits (not bytes), bitOrder
- streaming
- non-8-bit-characters (7, 6, 5)
- unbounded lookahead - parsing/backtracking, and unparsing

Written in Scala

Daffodil Code Base



Data as of 2018-08-06

Daffodil Integrations

- Apache
  - Spark
  - NiFi
  - ….
    - If you can intake/export XML or JSON, then Daffodil enables you to handle anything else describable with DFDL.
- Non-Apache
  - XProc - Calabash XML Pipeline Engine
  - Software^AG webMethods™ Integration Server

Demonstration

Apache Spark Integration ++

As we saw….

- Works Today for XML, JSON

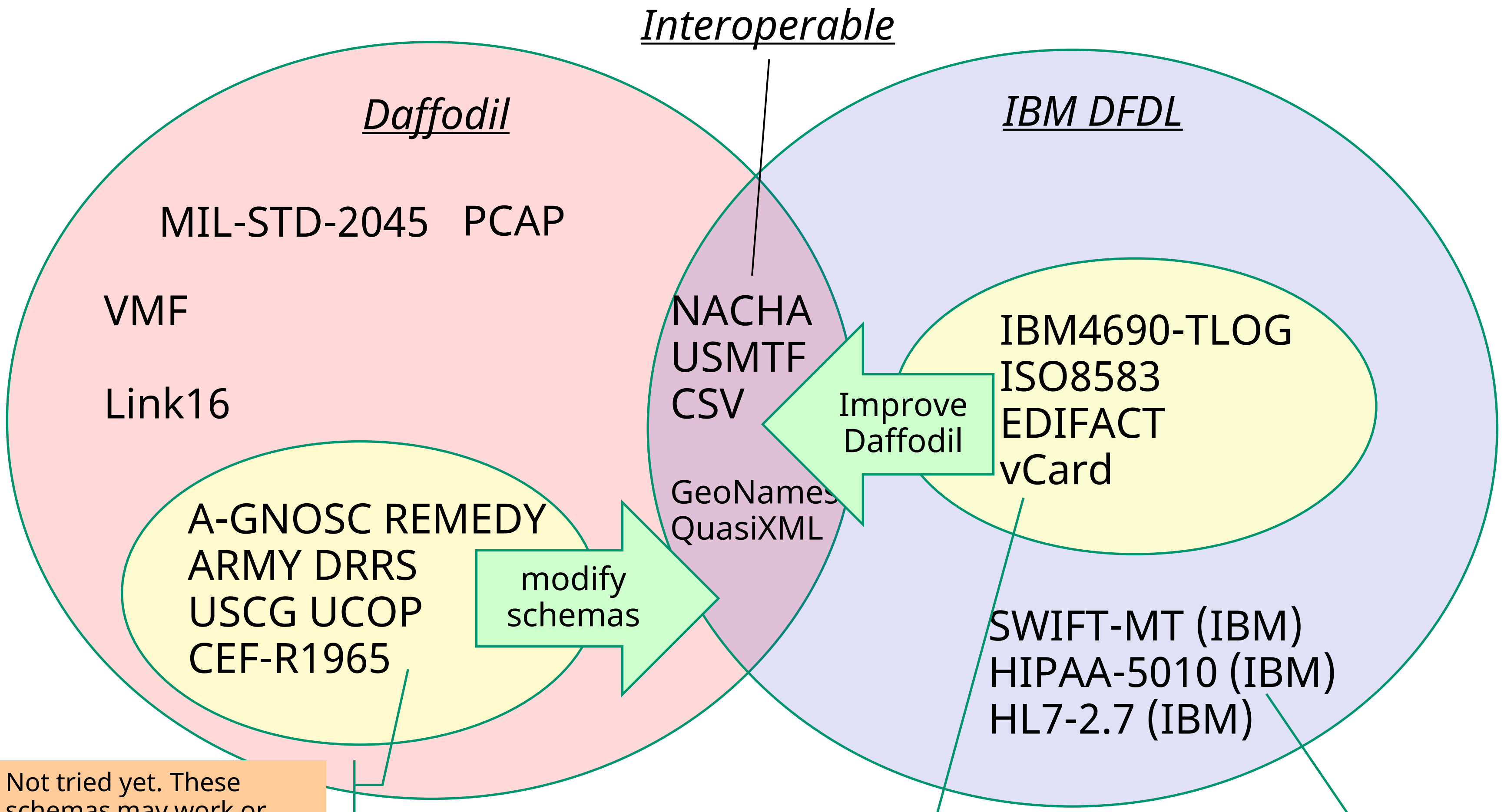
In the future….

- Spark Struct and DFDL Schema are very similar in logical data capabilities
  - DFDL Schema subset of XSD leaves out all the XML Schema "markup language" aspects
- Adapter could tightly integrate DFDL schemas with Spark Struct
  - Directly Construct/Consume Struct Metadata
  - Directly Populate/Consume Spark Struct Instances
- Avoid overheads (speed, and intellectual baggage) of XML/JSON conversion

Daffodil Future Development

- Cross-validation/test with IBM DFDL
  - Interop test on all IBM-created DFDL schemas
- Tutorials on writing/debugging DFDL schemas
- Improved trace and debug
- Full SAX-style streaming behavior for parsing, unparsing
- Faster schema compilation for large schemas
- Integrate into more frameworks
- Extensions: recursion, BLOB/CLOB, table-lookup,…

Interoperability – Daffodil & IBM



Not tried yet. These schemas may work or require small changes to work on IBM DFDL.

EDI and FINSERVE formats known to require additional DFDL features in Daffodil.

Schemas available under license from IBM; need permission to test.

Why is DFDL Needed?

There are *hundreds* of ad-hoc data format description systems

Every Enterprise Software Company

- IBM (10+)
- Oracle(10
- SAP(10+)
- Microsof
- SAS
- Informat
- SyncSort
- AbInitio
- Pervasive
- Qlik/Expr
- Pentaho
- .... Dozens mo

Every kind of software that takes in data:

kers)

Even within products of the same company!

Software world is very different now.

All this proliferation of redundant engineering can be avoided in the future.

DFDL Language Standard + 300K lines of Apache Licensed Daffodil can eliminate this data format problem ***finally***.

Collaborators Needed

- Apache Daffodil (Incubating)
  - Growing our Community
    - To graduate from incubator to full Apache project status *requires committers from more organizations*
    - Scala programmers wanted
- DFDL Schemas
  - Join github DFDLSchemas community

Questions?

DFDL Specification:

http://ogf.org/dfdl

DFDL Schemas:

https://github.com/DFDLSchemas

Daffodil Open Source:

https://daffodil.apache.org

users@daffodil.apache.org

dev@daffodil.apache.org

Twitter @ApacheDaffodil

mbeckerle@tresys.com
mbeckerle@apache.org

Twitter @Tresys

Open DFDL Examples/Integrations

https://github.com/OpenDFDL

Java helloworld for DFDL

daffodil-spark example code ❤ ❤ ❤

END

Extra slides may follow for use in optional discussion.