A Language for Engineering Design

Walter W. Wilson

Lockheed Martin Enterprise Information Systems & The University of Texas at Arlington

Workshop on Product Data Exchange
May 2-4, 2007
Santa Barbara, CA
Overview

What is the best way to represent an engineering design?

• Problems with commercial CAD systems
• Limitations of STEP
• Proposal:
  1. A language for engineering design
  2. Open source geometric engine
  3. A minimal standard language for implementation
• Related work
• Summary
Problems with CAD Systems

- Data stored in a proprietary binary format
- You need a license to access your own data!
- Hard to transfer high-level info between systems
- Will data be accessible decades from now?
- Can we have the openness of paper drawings with a digital representation?
Limitations of STEP

• useful operations & geometry types missing:
  – geodesic curves, tapered offset surfaces

• lags behind commercial CAD systems
  – standardization process is slow
  – must compromise differences in systems
  – vendors may be slow to implement

• no programming capability
  – a data-only format is inherently limited in its features
  (programmability gives extensibility without having to change the standard)
  – customizations not supported

Thus, information usually lost when writing STEP.
Proposal

1. A language for engineering design
2. Open source geometric engine for this language
3. Implementation of geometric engine in a standard minimal declarative language
An Engineering Design Language - EDL

- Textual, human-readable language for engineering design
- Stores definitions of geometry instead of megabytes of data
- Screen image, STEP files, etc., generated from the text definitions
- Interactive system would create & edit the text files
- A domain-specific language for "complex" data (functions + data)
EDL Example

(definations
...
(PT3 (coords 5 15 1.5))
(CRV1 (cubic-spline PT1 PT3 PT4 (project PT2 SURF1)))
(SURF2 (offset 0.1 SURF1))
(CRV2 (geodesic SURF2 PT1 PT4)) ! geodesic curve
(ht .5) ! symbolic constants
(radius .3)
(depth .2)
(SOL1 (cuboid (coords 0 0 0) (coords 2 3 ht))
   (hole (coords 1 1.5 ht) radius depth))
! – cuboid solid with cylindrical hole
...
)
EDL Example (2)

Language would support new function definitions:

(function (Block side)
    (height .5)
    (- (cuboid (coords -side/2 -side/2 0)
                (coords side/2 side/2 height))
       (+ (cylinder (coords -.5 -.5 0) (dir 0 0 1) .25)
          (cylinder (coords .5 .5 0) (dir 0 0 1) .25))
    ) )

(Block1 (Block 2.0))
(Block2 (Block 2.5))

... or

(for i in 0..9 do
    (B<i> (Block 2.0+.25*i))
)
EDL Advantages

• Text definitions are human readable
• Data is stored at a higher semantic level
  – Engineering knowledge can be captured
• Text definitions complement image
  – Image gives fast, intuitive understanding
  – Text gives precise and complete information
• Editing can be done both textually & graphically
  – Some modifications are easier in text: undo/redo
• Complete history and associativity is inherent
• Parametric design is encouraged
EDL Advantages (2)

• Definition of new functions is easier
  – Customizations can be saved
• Better support for design automation
  – Creating new functions for repetitive operations
  – Adding analysis functions for design optimization
• Comments can be added to definitions
• Definitions would be “exact”
  – 0.1, 1/3, cos30deg, geodesic
• Easier design reuse
• Data has modeler independence
The Case for an Open Source Modeler

- Better algorithms
- Enables continuous improvement of EDL
- Understanding and control of approximations
- Identical results on different systems
- Long-term accessibility
- Implementation openness
The Language for EDL Implementation

• this would be the standard – not the EDL
  – allows engineering design language to evolve
• requirements:
  – small & elegant
  – long-lasting (forever?)
  – explicit approximate arithmetic
    • symbolically defined floating point operations
    • identical results down to the last bit
  – efficiency not a requirement for long-term archiving
  – efficiency may not be a requirement for general use if sophisticated optimization is possible
• use minimal functional or logic programming language
My Pick for EDL Implementation

• "axiomatic language" – www.axiomaticlanguage.org
• specification language
  – functions defined without implementation algorithm
• minimal – nothing is built-in
  – easy to standardize
  – approximate arithmetic symbolically defined
• meta-language
  – able to define other languages within itself
  – EDL would be an embedded domain specific language

But automatically transforming specifications to efficient algorithms is an unsolved problem!
Related Work

• XML – Ok, but not a programming language
• Programming APIs – Djinn
• Domain specific languages – EREP, PLaSM
• Embedded DSLs
  – Haskell.org (graphics, animation, music)
Summary

• A language is a better representation for engineering design data
• A better design product
  – Higher level definitions
  – Better documentation and understanding
  – More optimal design
• Greater productivity for the engineer
  – Easier refinement of the design
  – Increased design automation
• Open source definition of EDL is essential
• "axiomatic language" is good candidate
Appendix – Axiomatic Language Semantics

Axioms generate valid expressions.

**expression:**
- an **atom** – a primitive, indivisible element,
- an **expression variable**, or
- a **sequence** of $\geq 0$ expressions and **string variables**

**axiom** – a **conclusion** expression and $\geq 0$ **condition** expressions

Axioms generate **axiom instances** by substituting values for the expression and string variables. (expressions for expression variables; strings of expressions and string variables for string variables)

**valid expression** – If all the conditions of an axiom instance are valid expressions, then the conclusion is a valid expression.

Valid expressions are interpreted as functions and programs.