A Programming Language Approach to Parametric CAD Data Exchange

*Presented by:* John Altidor
Jack Wileden, Jeffrey McPherson,
Ian Grosse, Sundar Krishnamurty,
Felicia Cordeiro, Audrey Lee-St. John
Problems in CAD Exchange

- Data exchange between different computer-aided design (CAD) systems is a major problem inhibiting integration.
- Existing standard CAD formats (e.g. IGES, STEP) are inadequate for:
  - Modification
  - Extension
  - Higher-level functionality
- Exchanging features generally requires manually recreating a CAD model.
Features Lost

IGES File Format

- Geometric information maintained
- Features cannot be selected
- Design intent gone
- In this example, pitch (number of threads) cannot be changed

NSF I/UCRC Center for e-Design
Related Work for Preserving Features

- **Commercial software for CAD translations.**
  - Proficiency
  - Translation Technologies
  - Elysium

- **Ontological approaches to CAD translations.**
  - Patil et al. “Ontology-based exchanged of product design semantics”.

- **See paper for comparison.**
Language Approach

- CAD systems are modeled as **programming languages**.
- CAD models correspond to **programs** in the languages modeling the CAD systems.
- Case studies applying our models, methods and tools to popular CAD systems (**Pro/Engineer** and **SolidWorks**) to assess and guide our research.
CAD Systems Background

1. Create a 2-dimensional (2D) circle with radius of 2cm.
2. Extrude the circle with depth of 4cm.
3. Create a cube with side-length 3cm.
4. Add a constraint to make centers of the cylinder and cube be separated by exactly 6cm.
CAD System as Languages

**Category**
- **Model**
  \[ M ::= S \mid \text{Sequence}(S, M) \]
- **Statement**
  \[ S ::= F \mid K \mid \text{Assign}(x, F) \]
- **Feature**
  \[ F ::= n \mid x \mid \text{Extrude}(F_1, F_2) \]
  - Cube(F)
  - Circle(F)
  - Triangle(F_1, F_2, F_3) | ...

**Constraint**
- **K**
  \[ K ::= \text{Boundary}(F_1, F_2) \]
  \[ \text{MaxDist}(F_1, F_2, F_3) \]

---

- Cylinder
- Cube
- Circle
Conversion Strategy

Pro/Engineer
Native Format

Pro/Engineer
XML Schema

SolidWorks
Native Format

SolidWorks
XML Schema
Implemented

- Proof of concept for automatic conversion of 2D sketches.
Pro/E 2D Section
Generated XML of Pro/E Section

<pro2dsection name="S2D0001">
  <pro2dEntities>
    <pro2dEntity id="4" isProjection="false" type="PRO_2D_LINE">
      <end1><Pro2dPnt x="0.00" y="0.00" /></end1>
      <end2><Pro2dPnt x="200.00" y="0.00" /></end2>
    </pro2dEntity>
    <pro2dEntity id="5" isProjection="false" type="PRO_2D_LINE">
      <end1><Pro2dPnt x="200.00" y="0.00" /></end1>
      <end2><Pro2dPnt x="200.00" y="100.00" /></end2>
    </pro2dEntity>
    <!-- ... -->
  </pro2dEntities>
  <pro2dConstraints><!-- ... --></pro2dConstraints>
  <pro2dDimensions><!-- ... --></pro2dDimensions>
</pro2dsection>
<sw2DSection name="S2D0001">
  <sw2DEntities>
    <sw2DEntity ID="(4,0)" type="swSketchLINE">
      <Start><sw2DPt ID="(4,1)" x="0.0" y="0.0" z="0.0" /></Start>
      <End><sw2DPt ID="(4,2)" x="200.0" y="0.0" z="0.0" /></End>
    </sw2DEntity>
    <sw2DEntity ID="(5,0)" type="swSketchLINE">
      <Start><sw2DPt ID="(5,1)" x="200.0" y="0.0" z="0.0" /></Start>
      <End><sw2DPt ID="(5,2)" x="200.0" y="100.0" z="0.0" /></End>
    </sw2DEntity>
    <!-- ... -->
  </sw2DEntities>
  <sw2DConstraints><!-- ... -->
  <sw2DDimensions><!-- ... -->
</sw2DSection>
SW 2D Section from XML
Conversion Strategy

Pro/Engineer Native Format

Pro/Engineer XML Schema

Conversion logic middleware

SolidWorks Native Format

SolidWorks XML Schema

Pro/Engineer Native Format

SolidWorks Native Format

Conversion logic middleware
Conversion Logic Middleware

Pro/Engineer XML Schema

ProLang

Subset of 2D

Conversion semantics

SolidWorks XML Schema

SWLang

Subset of 2D
**ProLang Syntax** (Fragment)

<table>
<thead>
<tr>
<th>Category</th>
<th>AST Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Section}^P )</td>
<td>( S^P : \text{=} \text{Pro2D}(E^P, C^P, D^P) )</td>
</tr>
<tr>
<td>( \text{Entity}^P )</td>
<td>( E^P : \text{=} L^P \mid Q^P )</td>
</tr>
<tr>
<td>( \text{Line}^P )</td>
<td>( L^P : \text{=} \text{Line}^P(id^P, b, P_1^P, P_2^P) )</td>
</tr>
<tr>
<td>( \text{EntityPoint} )</td>
<td>( Q^P : \text{=} \text{EntPoint}(id^P, P^P) )</td>
</tr>
<tr>
<td>( \text{SimplePoint}^P )</td>
<td>( P^P : \text{=} \text{Point}^P(rx, ry, id^P) )</td>
</tr>
<tr>
<td>( \text{Constraint}^P )</td>
<td>( C^P : \text{=} \text{SamePoint}^P(id^P, P_1^P, P_2^P) )</td>
</tr>
<tr>
<td></td>
<td>( \mid \text{PntOnEnt}^P(id^P, L^P, P^P) )</td>
</tr>
<tr>
<td>( \text{Dimension}^P )</td>
<td>( D^P : \text{=} \text{LineDim}^P(id^P, r, L^P) )</td>
</tr>
<tr>
<td></td>
<td>( \mid \text{LinePointDim}^P(id^P, r, L^P, P^P) )</td>
</tr>
</tbody>
</table>

\[
\vec{A} = [A_1, A_2, \ldots, A_n], \text{ where } n \geq 0. 
\]
**SWLang Syntax (Fragment)**

<table>
<thead>
<tr>
<th>Category</th>
<th>AST Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Section}^S$</td>
<td>$S^S : = \text{SW2D}(E^S, C^S, D^S)$</td>
</tr>
<tr>
<td>$\text{Entity}^S$</td>
<td>$E^S : = L^S \mid P^S$</td>
</tr>
<tr>
<td>$\text{Line}^S$</td>
<td>$L^S : = \text{Line}^S(\text{ide}, P^S_1, P^S_2)$</td>
</tr>
<tr>
<td>$\text{Point}^S$</td>
<td>$P^S : = \text{Point}^S(\text{ide}, r_x, r_y, r_z)$</td>
</tr>
<tr>
<td>$\text{Constraint}^S$</td>
<td>$C^S : = \text{Coincident}^S(P^S_1, P^S_2)$</td>
</tr>
<tr>
<td></td>
<td>$\mid \text{HorizontalConstraint}^S(L^S)$</td>
</tr>
<tr>
<td>$\text{Dimension}^S$</td>
<td>$D^S : = \text{LineDim}^S(\text{idd}, r, E^S_1, E^S_2)$</td>
</tr>
<tr>
<td></td>
<td>$\mid \text{HorLineDim}^S(\text{idd}, r, E^S_1, E^S_2)$</td>
</tr>
</tbody>
</table>
Example Differences

- Points in ProLang and SWLang have 2 and 3 coordinates, respectively.
  - Point\(^P(r_x, r_y)
  - Point\(^S(idex, r_x, r_y, r_z)

- ProLang constraint PntOnEnt has no equivalent in SWLang.

- We converted a ProLang 2D section containing PntOnEnt constraints to an equivalent SWLang 2D section.
Conversion Semantics – Transition System

- Conversion semantics is a transition system on a set of states.
- A state is a pair: \((S^P, S^S)\)
- Transitions have designated starting states.

\[(\text{Pro2D}(\overline{E^P}, \overline{C^P}, \overline{D^P}), \text{SW2D}([\ ] , [\ ], [\ ]))\]

- Transitions systems complete when they reach one of their designated final states.

\[(\text{Pro2D}([\ ], [\ ], [\ ]), \text{SW2D}(\overline{E^S}, \overline{C^S}, \overline{D^S}))\]
Key Transition Rule

\[
\begin{align*}
S_2^P & \subseteq S_1^P & S_2^P & \equiv S_2^S \\
(S_1^P, S_1^S) & \mapsto (S_1^P - S_2^P, S_1^S + S_2^S)
\end{align*}
\]

- Premise \(S_2^P \subseteq S_1^P\) is needed to not introduce "new stuff."
Many-to-Many Mapping

\[
\text{intersect}(L_1^P, L_2^P) = P_{\text{int}}^P
\]

\begin{align*}
\text{Pro2D}([], [\text{PntOnEnt}^P(L_1^P, P^P), \text{PntOnEnt}^P(L_2^P, P^P)], []]) &= \text{SW2D}([P^S], [\text{Coincident}^S(\text{conv}_O(P_{\text{int}}^P), \text{conv}_O(P^P))], []])
\end{align*}

- The \text{intersect} function is a partial function that tries to compute the intersection point of two lines.
Summary – Theoretical Contributions

- We presented a rigorous model of CAD systems as programming languages.
- Formally defined algorithm for converting parametric CAD data between CAD systems: Pro/E and SolidWorks.
- We can convert some Pro/E 2D sections containing elements with no direct counterpart in SolidWorks.
Summary – Practical Contributions

- **Software** that automatically converts some Pro/E 2D sections to SolidWorks 2D sections.
- Open, text (XML) formats that allow users to write CAD models in native CAD systems’ formats **without using the GUI** of CAD systems.
- Other CAD interoperability researchers can apply their logic over our **open, easy-to-parse** formats to experiment with their approaches.
  - No need to learn CAD APIs.
Thank you!