Plug-and-Play Assembly Components

Jonathan Lund, Brett Black, C. Greg Jensen
Brigham Young University
brett.b.black@gmail.com, jon@jonlund.com
(801) 422-4996

Premium Partners:

Objective

- Objective: Be able to swap assembly components in and out and have them adapt intelligently to make sense in their new environment

- Solution:
  - Make part parameters depend upon their environment
  - Standardize interfaces to create ability to interchange with higher number of master models

Problem- How is knowledge reused?

- Original knowledge based engineering strongly based in parametric modeling
  - Morph a part to fit within a new role
- Limitations and challenges
  - Discrete features, topology changes
  - Assemblies—crossing the part boundaries

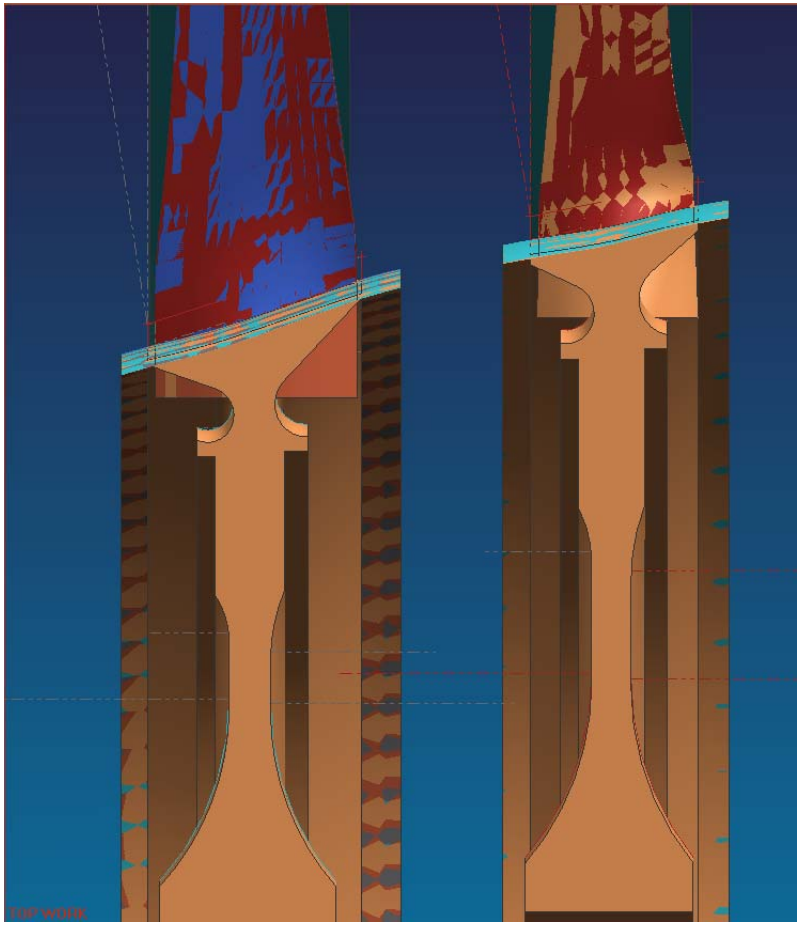
Traditional Modularity

- Equivalent to physical modularity
- Components can be swapped in and out
- Positioned, but not changed
  - (e.g. Bolts)

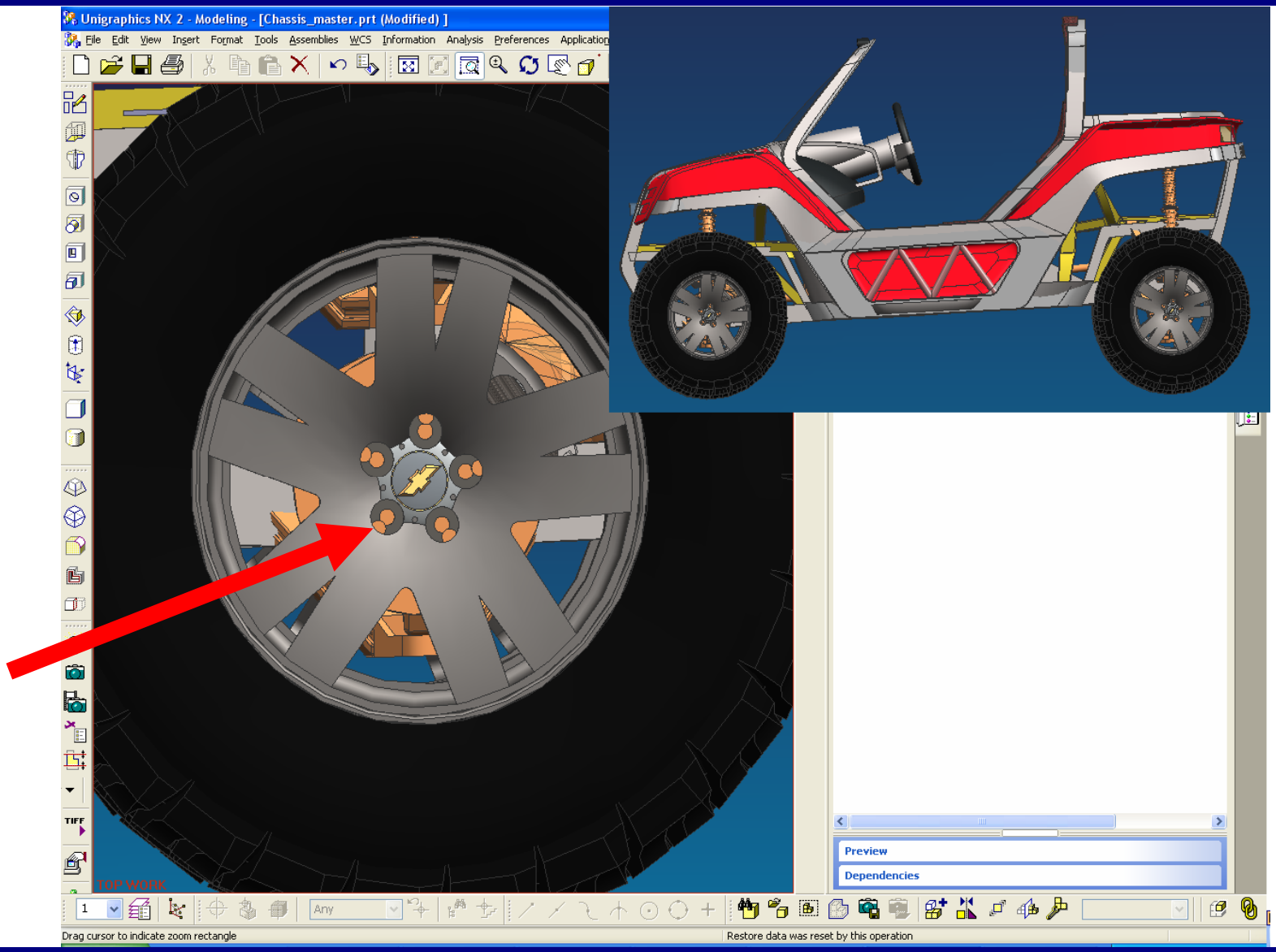
Traditional Modularity

Example: Jet Engine Compressor

- Assembly components commonly need to morph to fill assembly-specific roles
  - Example 1: compressor disks are nearly the same, but vary in size between stages
- Mating conditions are inadequate



Example: Wheel Rim



Adaptive Modularity

- Needs
  - Part positioning within assemblies
  - Features reacting within the part to the assembly *(Adaptive Modularity)*
  - Non-geometric information transfer
    - (e.g. Which stage?, What RPM?, What temperature?)
- Approaches
  - Interpart expressions
  - WAVE links
- Challenges (among many)
  - Interdependencies are often part/item-specific

Approaches

Externally-controlled parameters

- Master parameter list (Roach, 2003)
- External databases (Baker, 2005)
- XML
- Interdependent spreadsheets

Advantages	Disadvantages
Fast Robust updates Good for Optimization Familiar	Costly to develop and maintain No downstream associativity (parts contain no knowledge) Very little leverage of CAD system (interference detection, etc.)

WAVE– a Good Approach

- Company-specific configurators set WAVE links
  - Knowledge Fusion (i.e. GEAE SOLID)
  - UG Open
  - Manually
- Goal
  - Determine a robust general approach
    - Sensible conventions
    - Leverage Teamcenter Engineering
      - Product Structure Editor (PSE)

Background

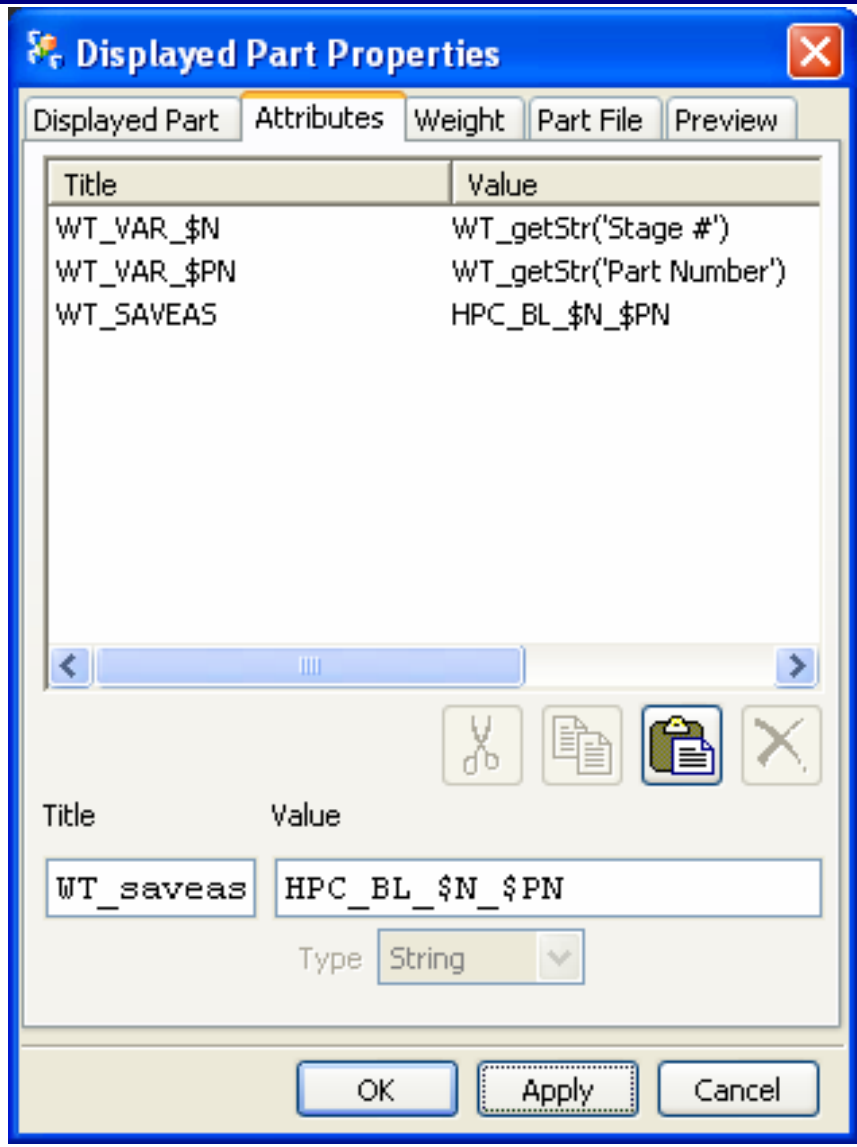
- WAVE links copy geometry from one part to another
- Each link requires three basic data:
  - Component (Part) with source geometry
  - Geometric object within this part to be copied
    - (i.e. sketch, line, face, etc.)
  - Destination component
- Reparenting WAVE links selects new data for first two items
  - This updates copied geometry and remains associative

AAC Assembly Preparation

- Create knowledgeable assembly
  - Create WAVE-linked models
    - Features depend on linked in geometry
  - Add generic attributes to link features
    - Source part (by coded name)
    - Source geometry (by name)
  - Save part with WAVE template convention
    - Append “_WT”

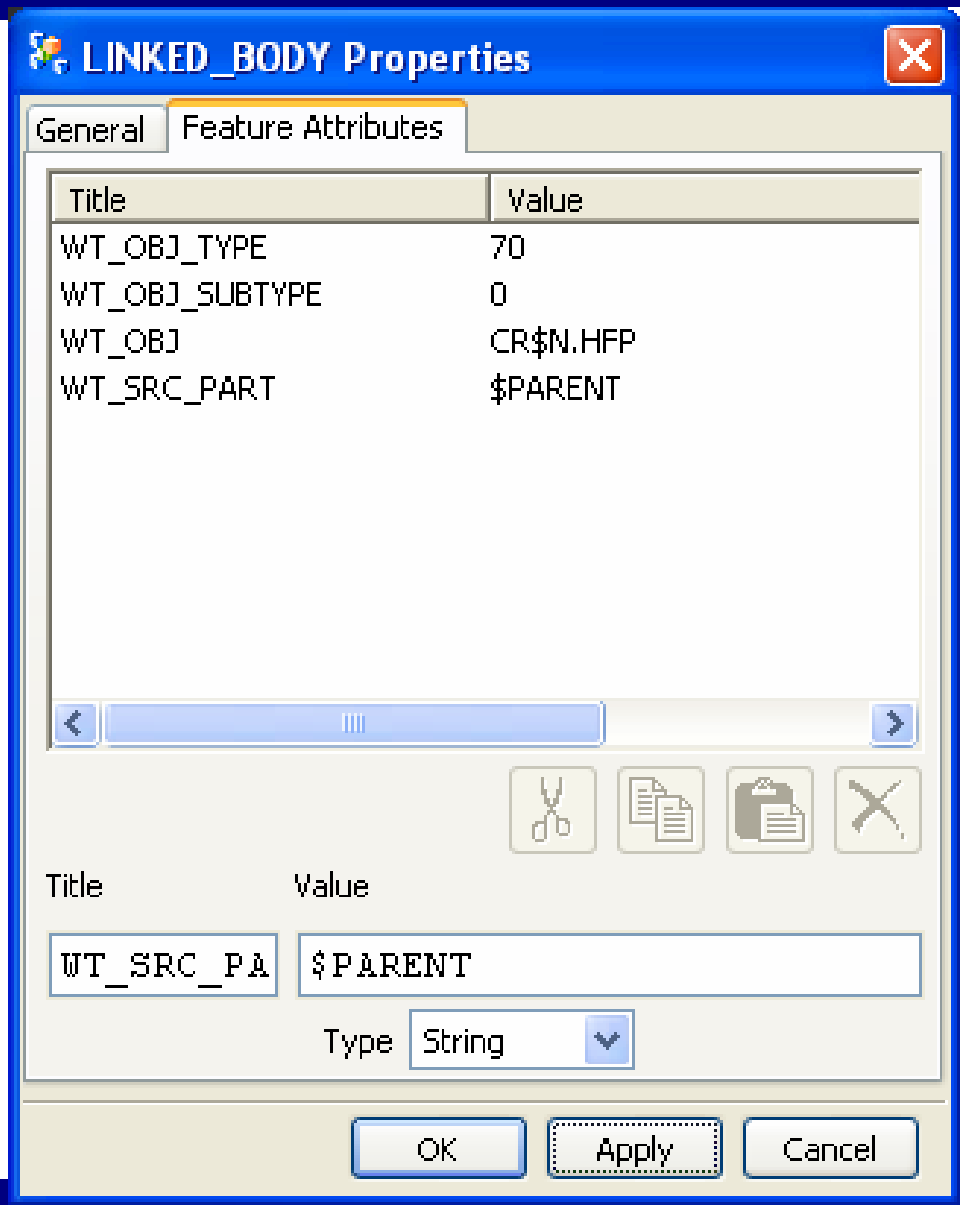
Conventions- Part

- Part attributes
  - Input Variable
    - Name: WT_VAR_$N (String)
    - Value: WT_getInt('Which stage is this?')
  - Mandatory Input Variable
    - Name: WT_VAR_$PN (String)
    - Value: WT_getInt(‘Enter part #’)
  - Interpreted Variable
    - Name: WT_SAVEAS (String)
    - Value: (example) “HPC_BL_$N_”
      - Original part name: “XXXXX_WT”
      - Interpreted to be: “HPC_BL_$N_$PN



Conventions- Link Features

- Feature Attributes
  - Source part
    - $PARENT
  - Source object
    - $N gets converted from part attributes
    - Also `WT_Select('','picture.bmp')`



Assembly Usage Structure

- UG/Open program does busywork
- Usage:
  - Add components to assembly
  - Position components to absolute zero
  - Run modularity tool
    - Identifies WAVE links
    - Queries them for reparenting information
    - Queries user for additional input as needed
      - Numeric, part, string, geometry
    - Reparents WAVE links
    - Queries for unique part number to replace “_WT”
    - Saves part uniquely with new name

Demo / Questions