Strategies For Working With Large Assemblies In I-DEAS

Brian Slick

Ferno-Washington, Inc.

bslick@ferno.com -or- brianslick@mac.com

800-733-3766

Premium Partners:

Brian Slick – I-DEAS History

- Co-op with SDRC, Fall 1996 (MS3)
- I-DEAS Instructor, June 1998 – October 2001
  - Design-related topics: Part Design, Assembly, 2D/3D Drafting, Best Practices, Surfacing, Harness, and C3P equivalents
- Contract Drafter → Sr. Project Engineer – Ferno
  - Created hundreds of parts, assemblies, drawings
  - “Assembly Manager” for several large (700+ instance) assemblies
  - User training and support
  - CAD evaluation and testing
- Winner of 2004 and 2005 PLM World “Top Gun” Contests

Ferno-Washington, Inc.

- Privately held, global company
- Multiple product lines in Emergency, Mortuary, Therapy, and Veterinary markets.

FERNO

Qualifiers

- Your mileage may vary. This presentation has been written based on procedures developed to meet the needs of Ferno. Techniques discussed may not always be “best” practices, and/or may not be applicable to every company’s needs.
- This presentation was created using I-DEAS 8m4, and re-verified in 10m4. Although the overall philosophy should still be applicable, changes in 11 or 12 may have obsoleted some of the techniques.

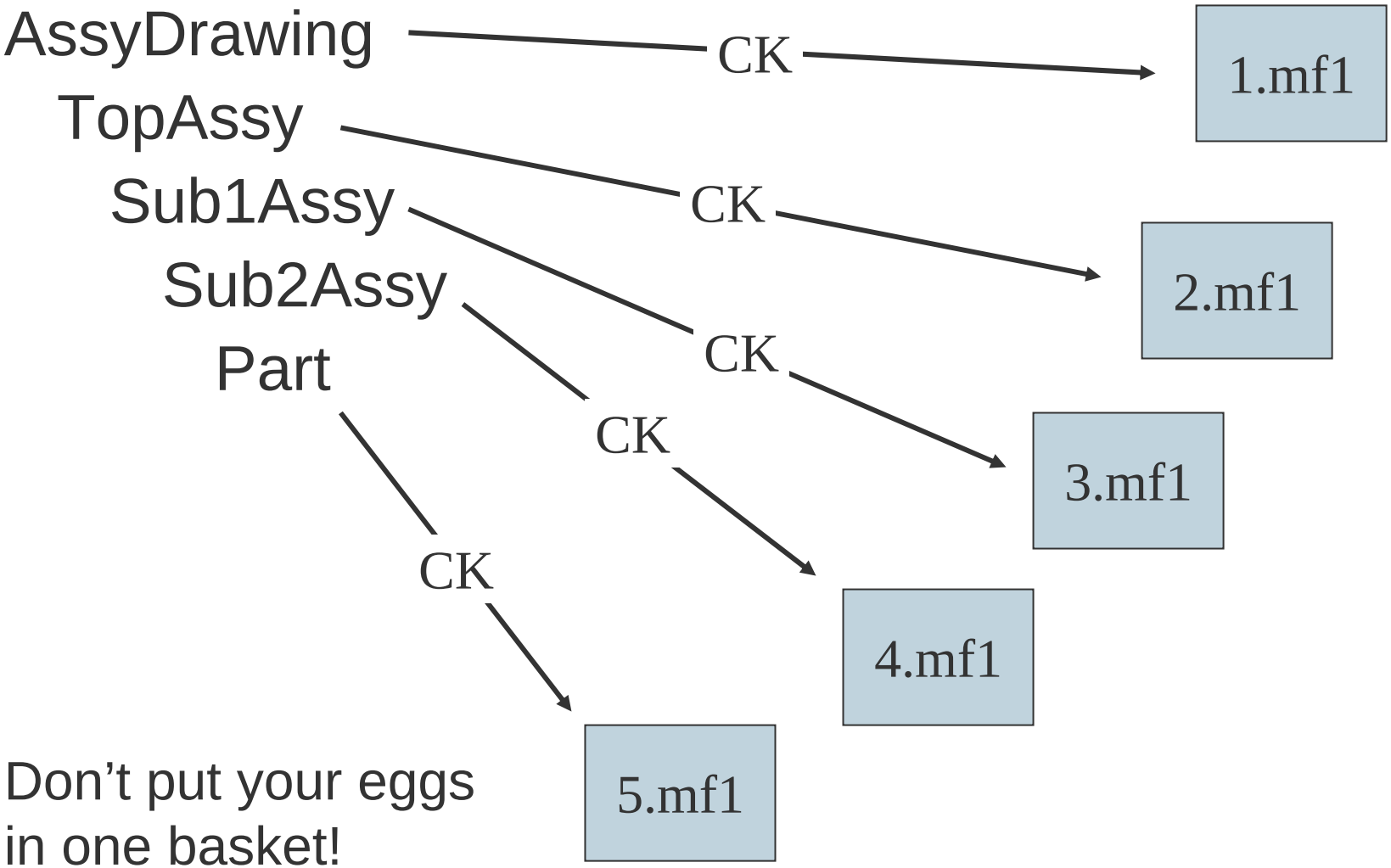
Overview

- Model File Usage
- Hierarchy Design
- Use of BORN Parts
- Constraining Tips
- Configuration Usage

Model File Deployment For Better Performance - 1

- Team: assign different assembly levels to different people
- Individual: Treat yourself like a 1 person “team” – use multiple model files as if you were multiple people
- Only check out what you need to work on. If you need to work on a single part, check out only that part. If you need to work on the top level, check out only the top level. Avoid having multiple levels checked out at once in a single model file.

Model File Deployment For Better Performance - 2



Don’t put your eggs in one basket!

Model File Deployment For Better Performance - 3

Cons:

- More library transactions
  - More library data (versions)
  - More idle time waiting for transactions

Pros:

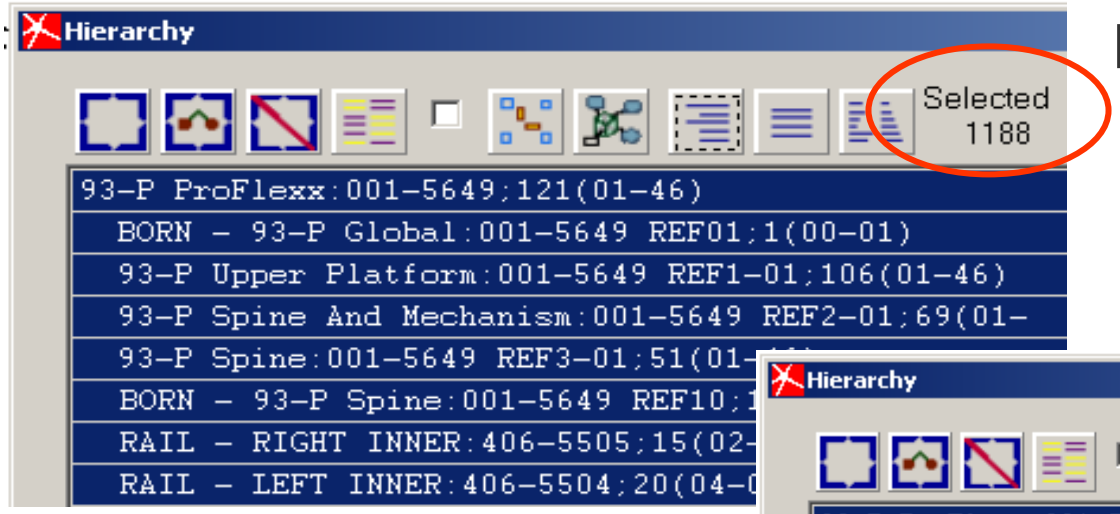
- Successful check-ins
  - Problems due to a sub-item being out of date are rare
- Better assembly solver performance
  - Cannot solve lower levels
- Easier to manage Update From Library
- Better buffering against corruption issues

Tip:

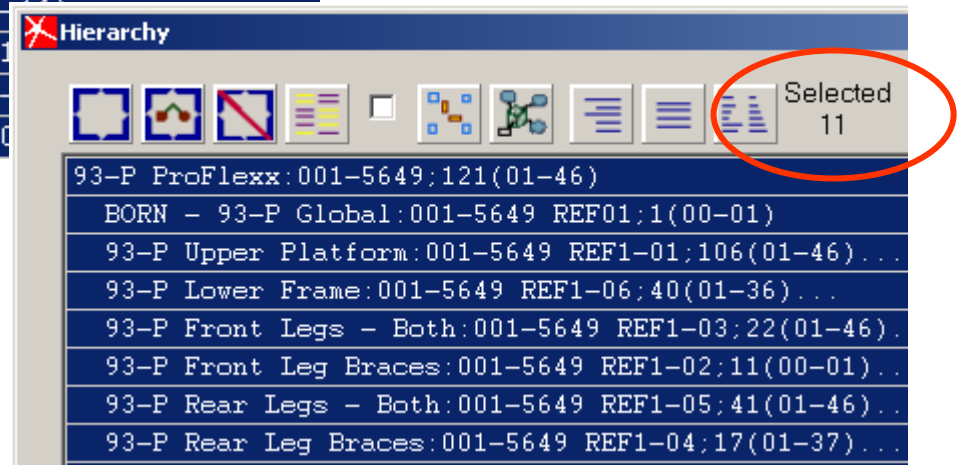
- When switching between large model files, restart I-DEAS to free up RAM.

Hierarchy Design - 1

- Use subassemblies!



Flat



Several Subassemblies

Hierarchy Design - 2

- The top level is going to require a lot of effort to solve no matter what you do – it's a large amount of data. In other words, be realistic and patient.
- Generous use of subassemblies can reduce (drastically!) the number of items the top level has to solve, thereby greatly improving solve performance.
- Small (50 instances or less) subassemblies are far easier to work with than 1 massive assembly.

Hierarchy Design - 3

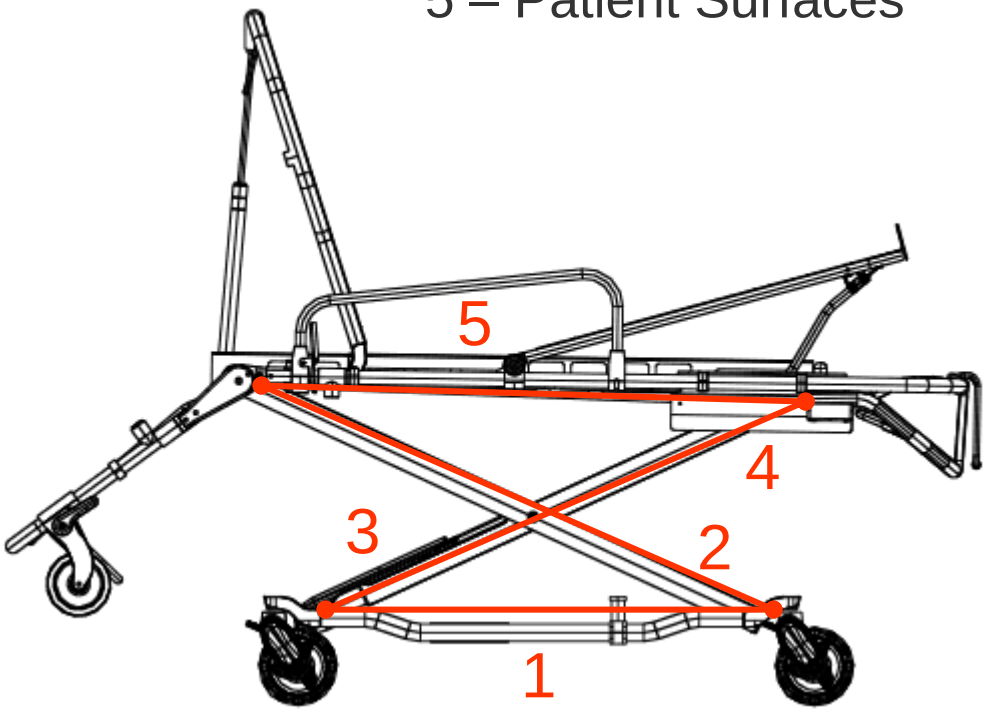
- Sometimes manufacturing needs will require subassemblies. These are “real” or on-the-shelf subassemblies, for example a weldment. Model in I-DEAS accordingly.
- User may need to create “fake” assemblies, solely for the purpose of making I-DEAS life easier. User has some leeway regarding the contents of these assemblies.
- Design “fake” assemblies to mimic real-life motion or real-world assembling method.

Hierarchy Design - 4

Minimum number of subassemblies should mimic minimum number of rigid bodies in a simplified mechanism.

A wireframe mechanism can be useful for planning an assembly hierarchy

1 – Lower Frame
2 – Fixed Length Legs
3 – Telescoping Legs #1
4 – Telescoping Legs #2
5 – Patient Surfaces

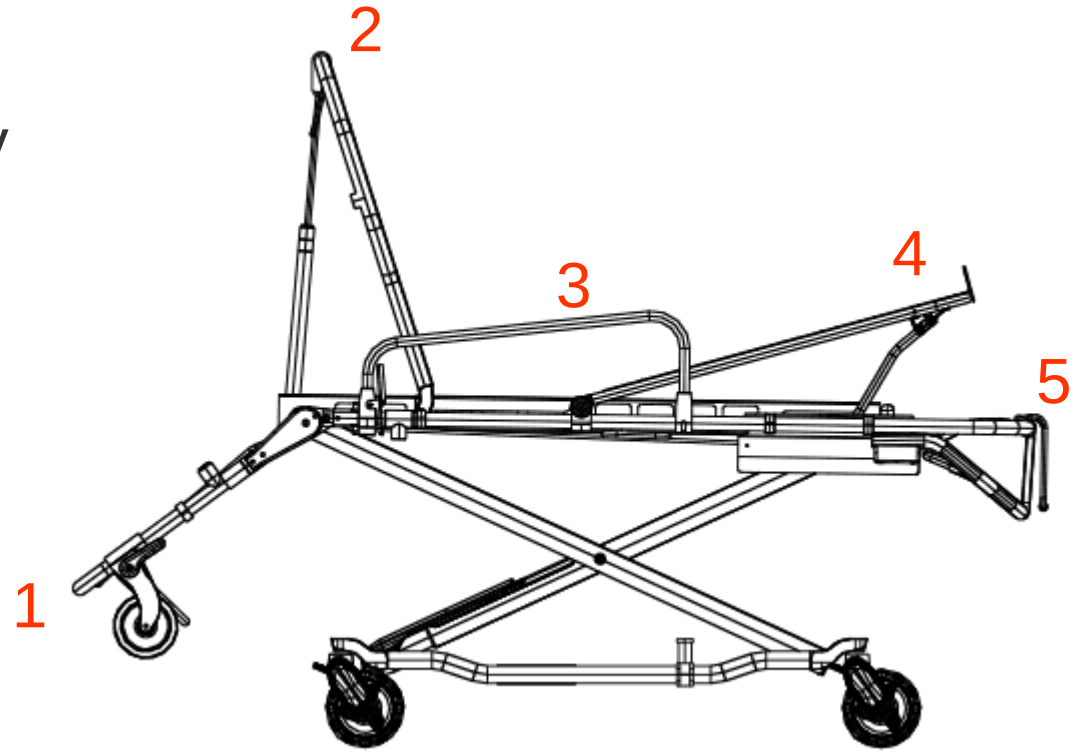


Hierarchy Design - 5

If 1st-level subassemblies can be further divided, do so.

Patient Surfaces Subassy

1 – Drop Frame
2 – Backrest
3 – Sidearm (x2)
4 – Shock Frame
5 – Main Frame

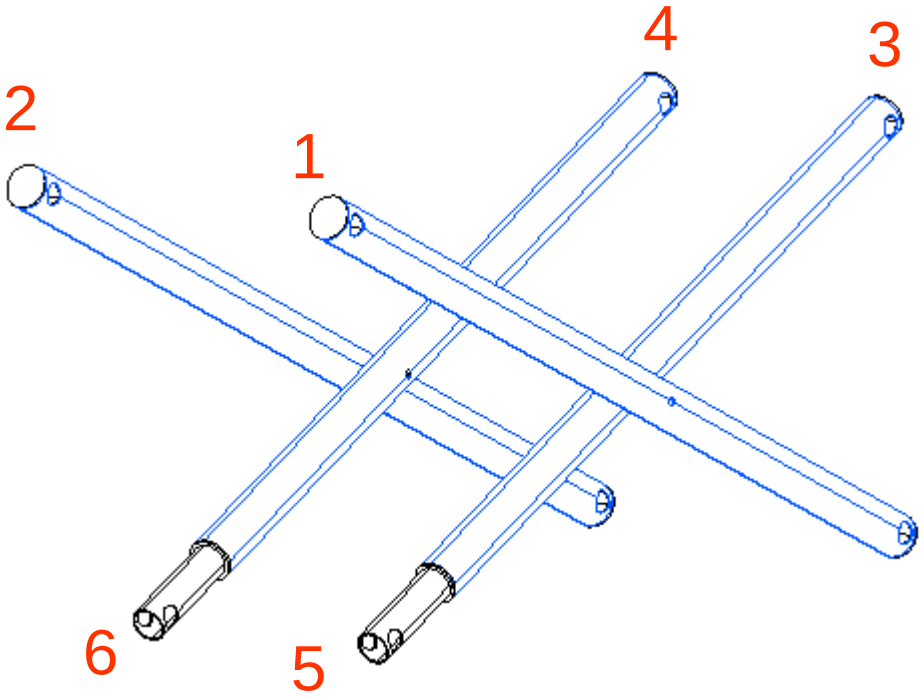


Hierarchy Design - 6

Symmetry/Pairing can provide performance benefits.

"Real" TopAssy

1 – Outer Leg R
2 – Outer Leg L
3 – Inner Leg Upper R
4 – Inner Leg Upper L
5 – Inner Leg Lower R
6 – Inner Leg Lower L



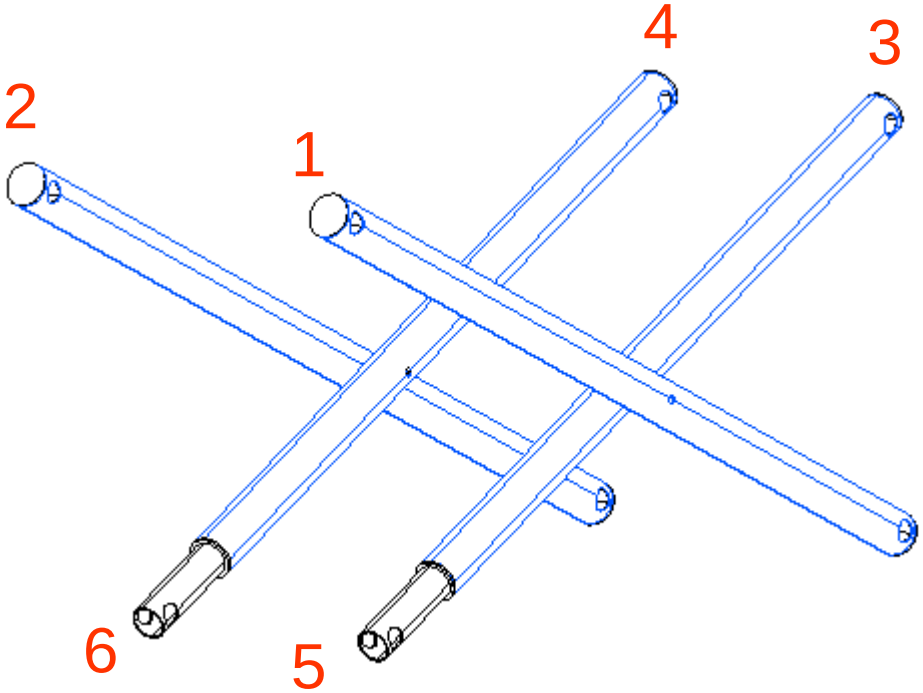
Hierarchy Design - 7

- In previous example:
  - One set of legs defines the mechanism, but the solver must calculate the position for both sets of legs.
  - Risk of a redundant or over-constrained assembly
  - Top level must solve positions for 6 items, and that's just for the legs. There are other 1st-level subassemblies as well.
- Recommended constraint method:
  - Constrain one set of legs to define the assembly mechanism/motion.
  - Constrain the other set to simply follow the first.

Hierarchy Design - 8

- Create more subassemblies for better performance.

“Fake” TopAssy
  Outer Legs
    1 – Outer Leg R
    2 – Outer Leg L
  Inner Upper Legs
    3 – Inner Leg Upper R
    4 – Inner Leg Upper L
  Inner Lower Legs
    5 – Inner Leg Lower R
    6 – Inner Leg Lower L



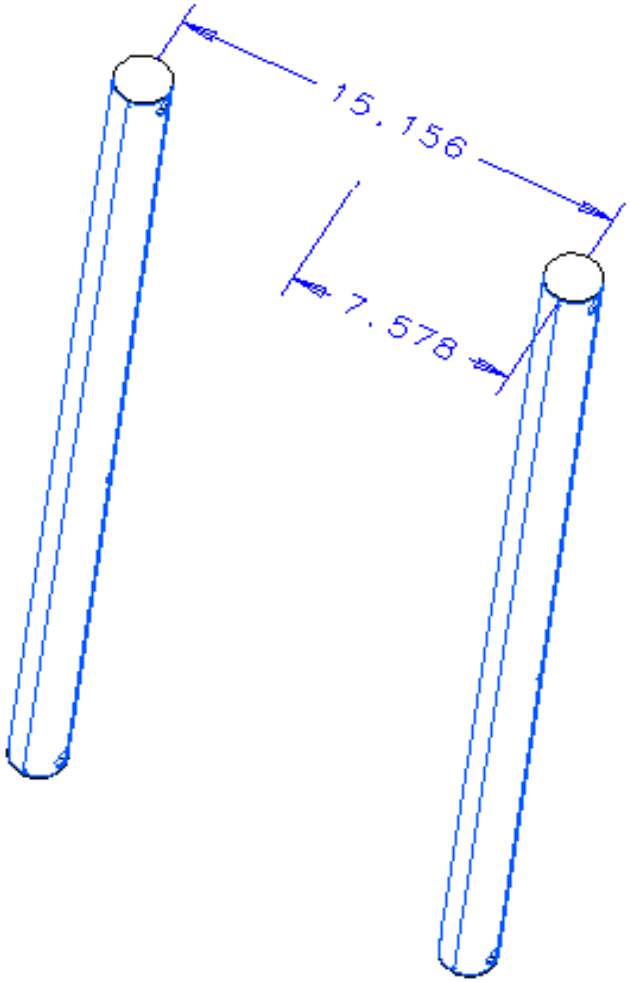
Hierarchy Design - 9

- In previous example:
  - Top level now only has 3 leg items to solve for.
  - No risk of a redundant or over constrained assembly
- Pros:
  - Fewer top-level constraints
  - Better top-level solver performance
- Cons:
  - Leg location side-to-side must be controlled at subassembly level, which isn’t realistic.

Hierarchy Design - 10

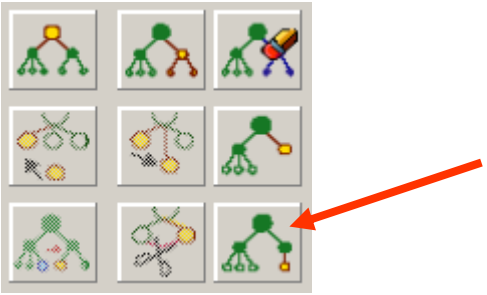
This constraint method probably does not represent a realistic assembly, since the legs are not related directly to each other in the real world.

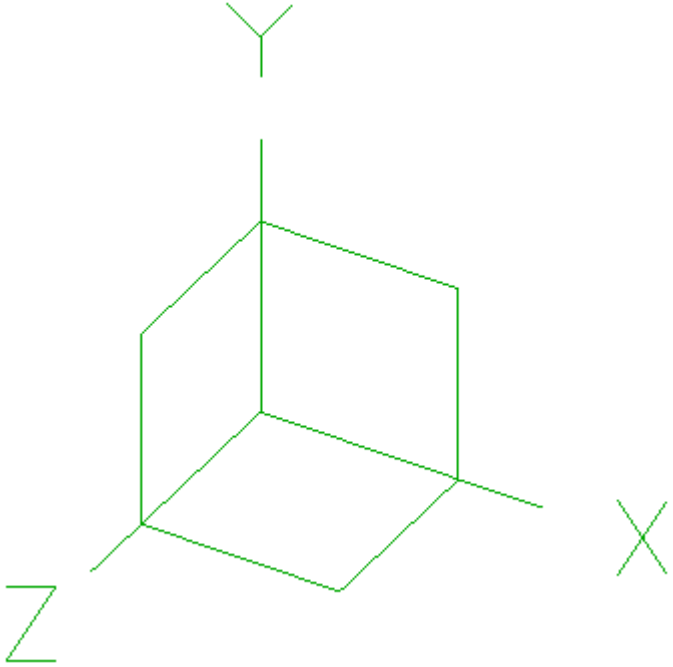
This is a trade-off between performance and realism. The sacrifice in realism may be unacceptable for some companies. For others, the performance gain will offset the sacrifice.



BORN Parts - 1

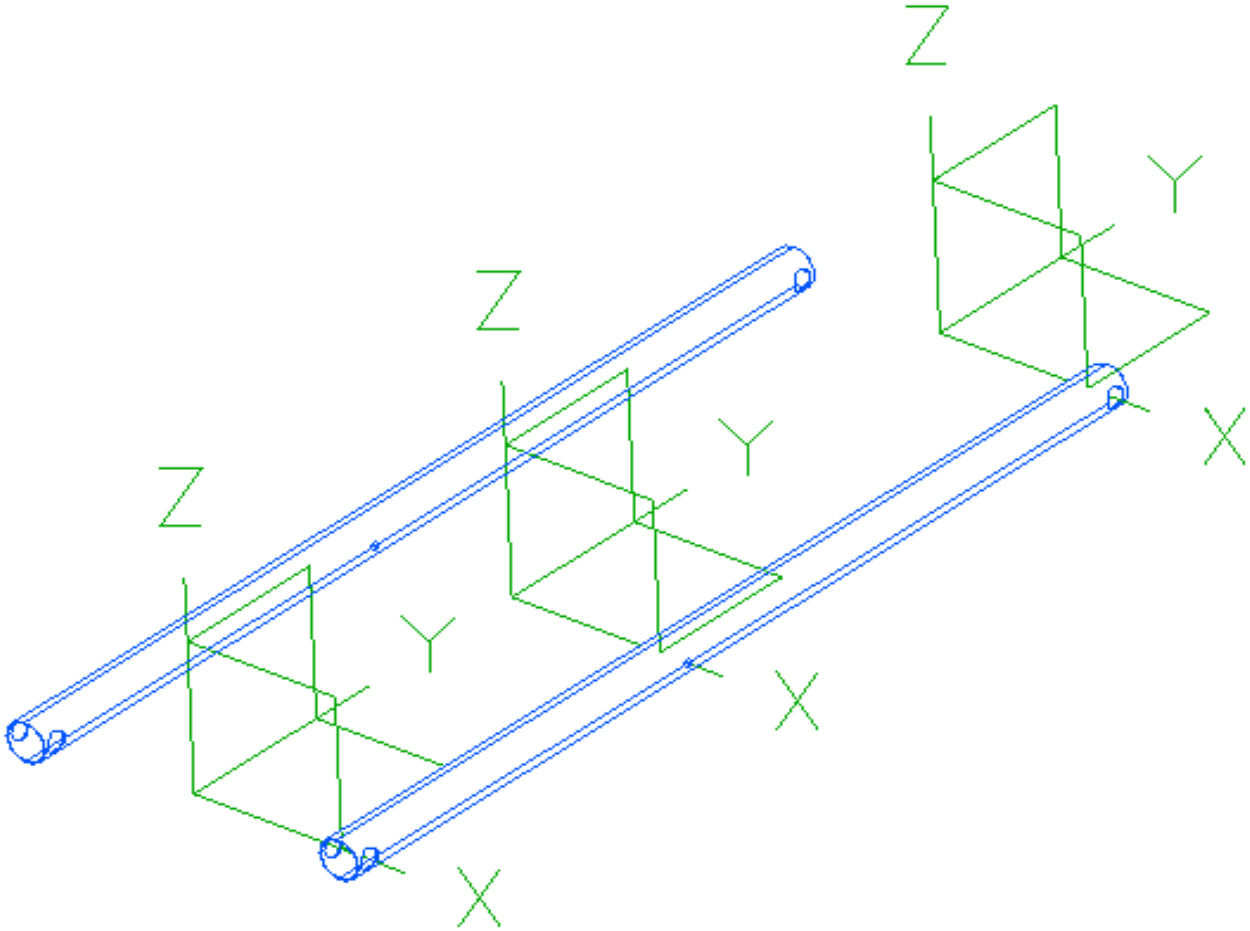
- Use Add Empty Part or any of the normal BORN creation methods
- Change color and size of CS to differentiate





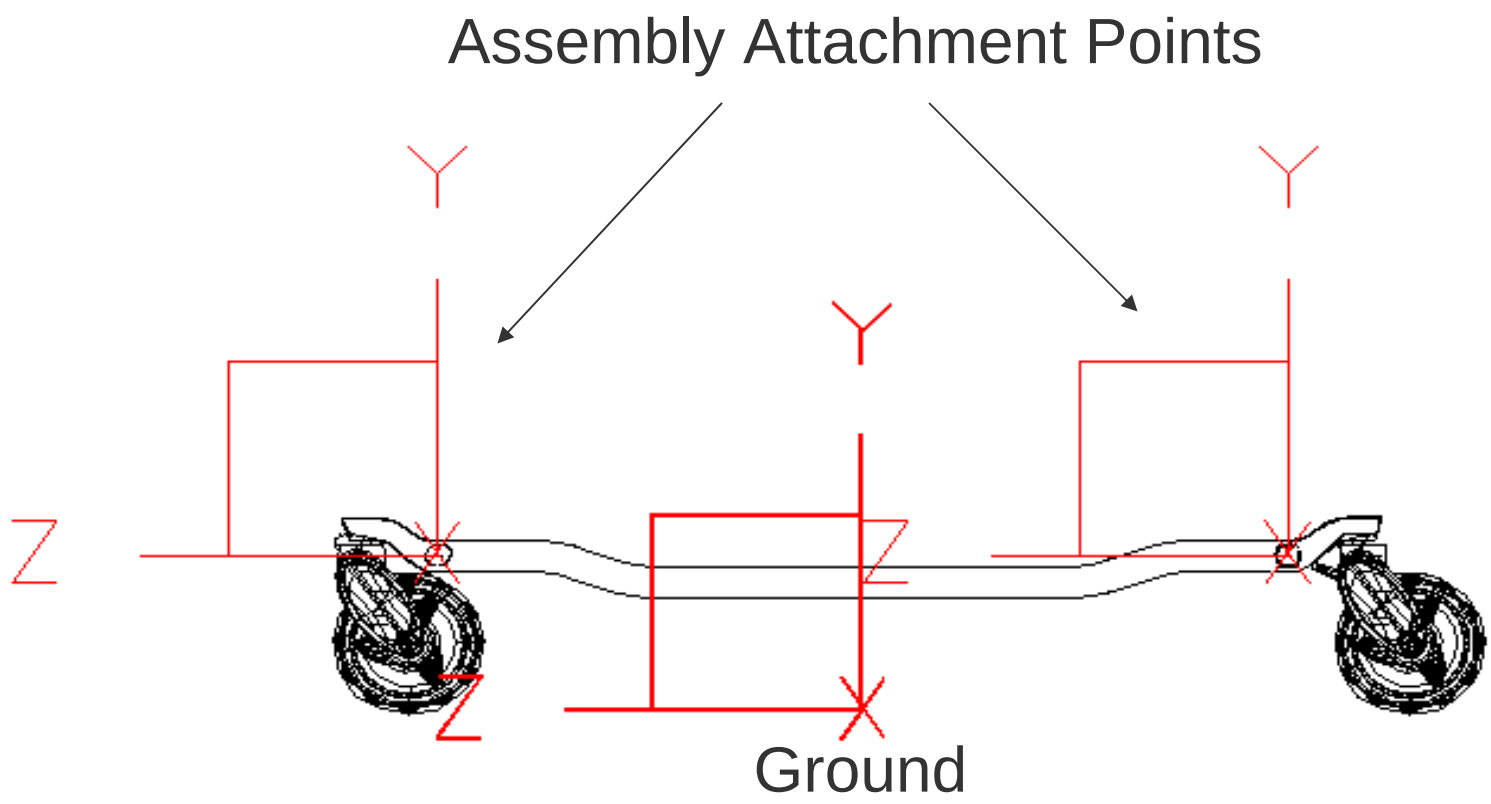
BORN Parts - 2

Constrain these BORNs to sit at important assembly attachment points within each subassembly.



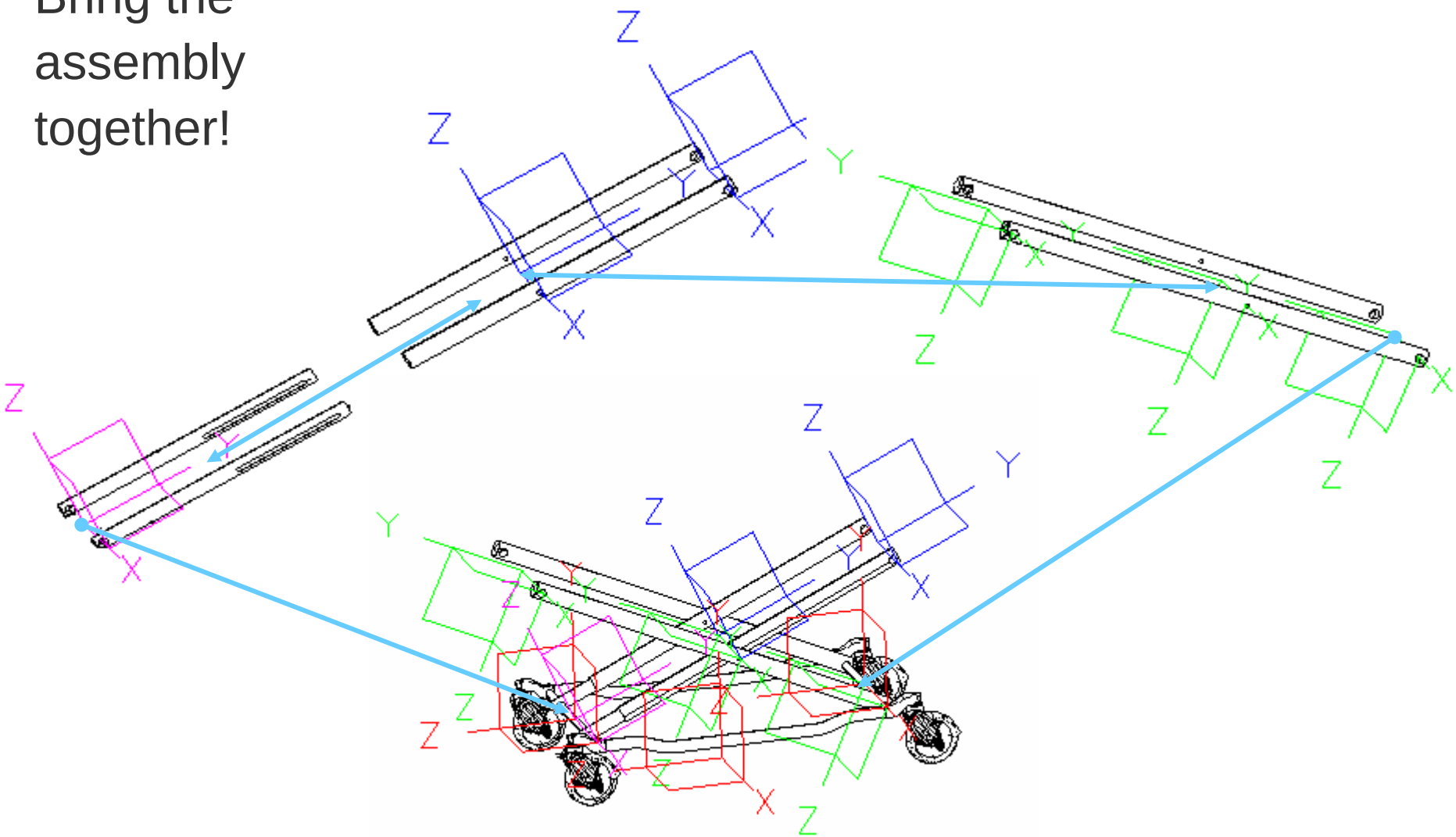
BORN Parts - 3

Use one BORN in the top level to serve as a visual ground/anchor.



BORN Parts - 4

Bring the assembly together!



BORN Parts - 5

Cons:

- Sacrifice some realism in the constraint method

Pros:

- Easier to constrain the top level. Just attach the BORNs to each other
- MUCH easier to track down the constraints that link the subassemblies together (hide everything except the BORNs)
- Provides a buffer against part changes or replacements
  - Since no parts are used directly to define top level relationships, no part change will ever wreck the top level assembly
  - Broken constraints are isolated in the subassembly, simply requiring the BORN to be reconstrained

BORN Parts - 6

Tips:

- A single BORN part can be reused in multiple places. However, having separate parts with descriptive names will assist in cases where multiple BORNs sit on top of each other
- Normal CS size is 0.15. Making a BORN part of 0.25 will typically suffice to stand out from the crowd
- A color not normally found in the standard color wheel will also assist with visibility
- A part name that identifies the part as a BORN (ex: BORN – Sidearm Pivot) will assist with filtering

Miscellaneous Statistics

- Largest Ferno I-DEAS assembly contains 1187 total instances. Here's a breakdown:
  - 453 unique parts/assemblies
  - 34 "fake" subassemblies
    - A typical fake assembly has fewer than 50 instances
  - 16 unique BORN parts
- The assembly is fully populated with hardware – nuts, bolts, washers, springs, etc.
- The assembly is <u>fully constrained</u> in a manner that permits animation

Usage Sample

The preceding practices required the development of a naming scheme to maintain some semblance of sanity. Here is Ferno's solution:

123-4567	Real Item
123-4567 REF##	BORN Part
123-4567 REF1-##	Fake 1st-level subassembly
123-4567 REF2-##	Fake 2nd-level subassembly

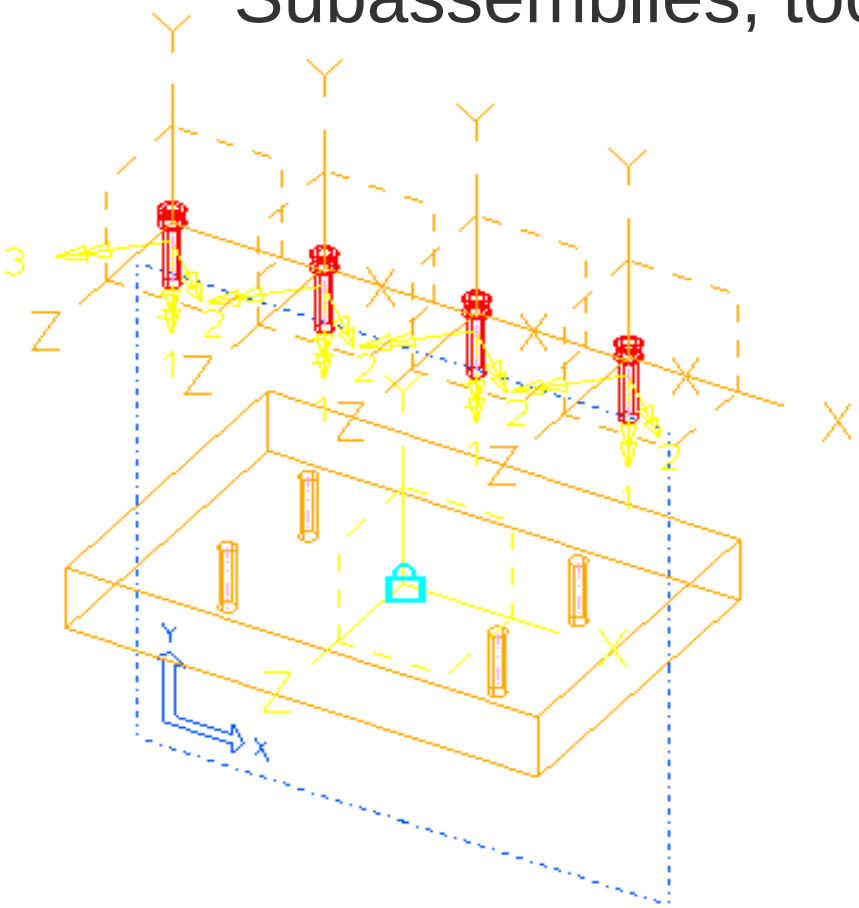
Descriptive names certainly help, as well as other visual cues such as changing case:

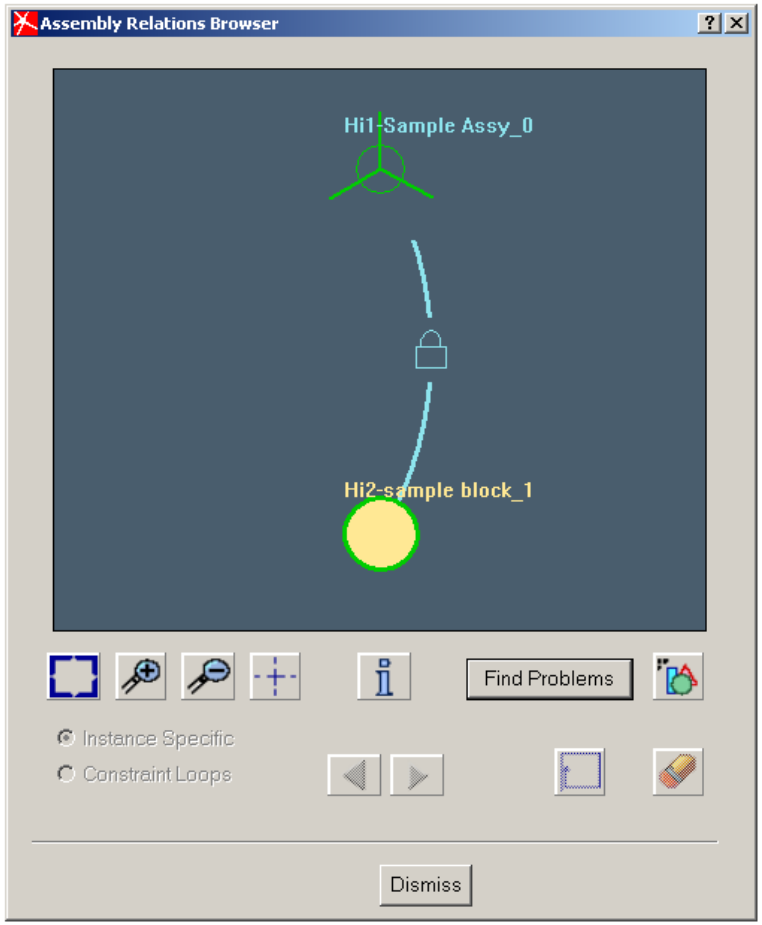
001-5620	35P COT – AMBULANCE PROFLEXX
001-5620 REF1-03	35P Both Non-Telescoping Legs
001-5620 REF2-05	35P Single Non-Telescoping Leg
001-5620 REF04	BORN – 35P Non-Telescoping Legs

Constraint Tips - 1

Create at least one lock for every assembly.

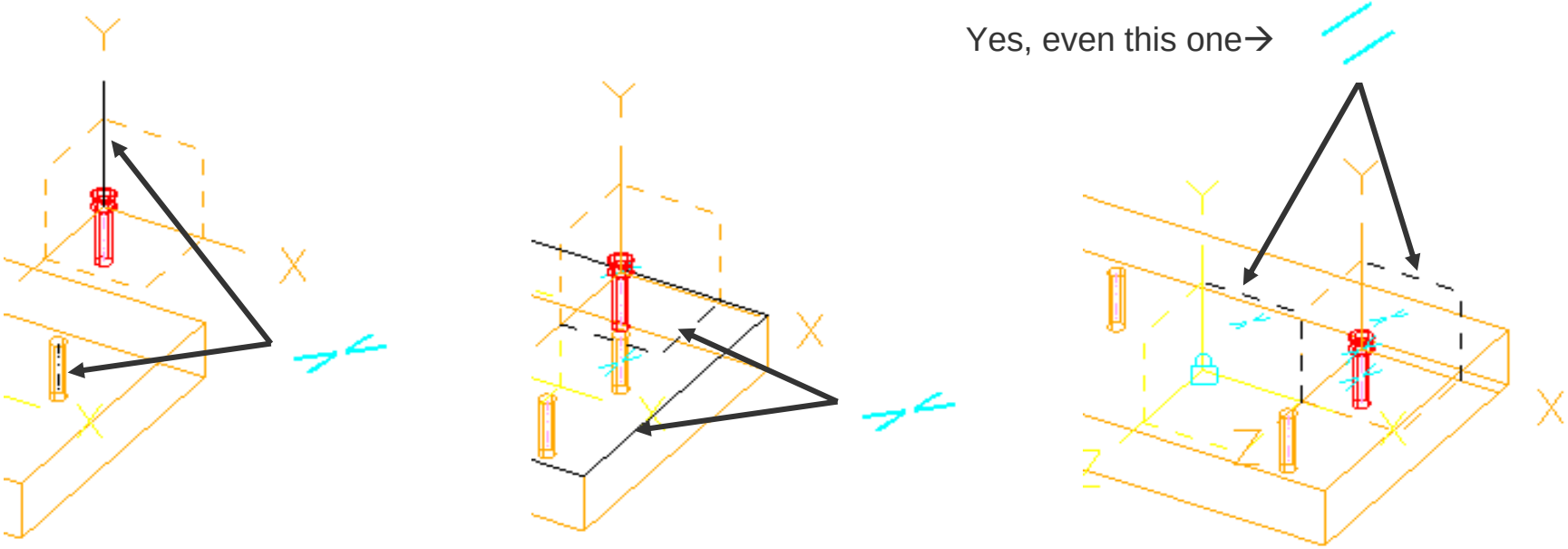
Subassemblies, too.





Constraint Tips - 2

Fully constrain all instances. Dangling DOF's can add up!



1 2 3

Configurations - 1

Configs Used =
Better performance

Manage Configurations

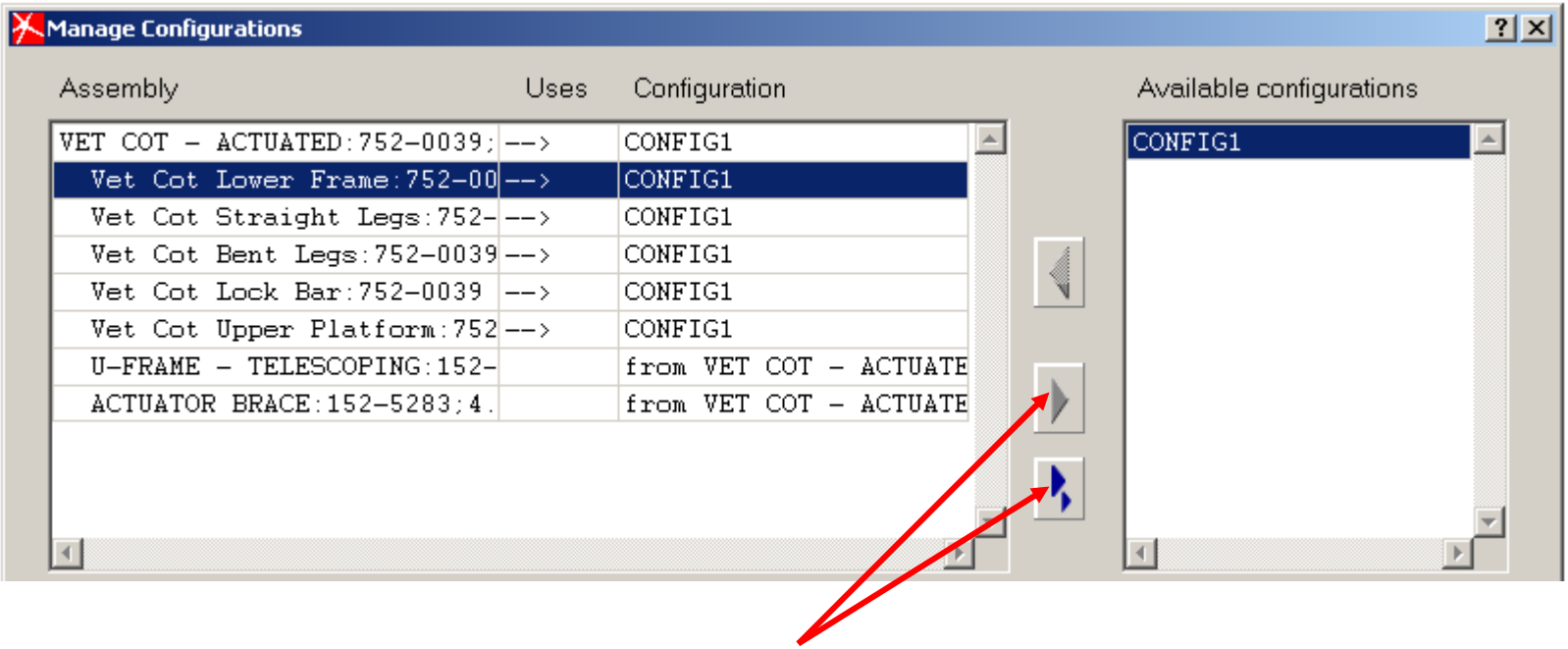
Assembly	Uses	Configuration
VET COT - ACTUATED:752-0039;	-->	MAX HEIGHT
Vet Cot Lower Frame:752-00	-->	CONFIG1
Vet Cot Straight Legs:752-	-->	CONFIG1
Vet Cot Bent Legs:752-0039	-->	CONFIG1
Vet Cot Lock Bar:752-0039	-->	CONFIG1
Vet Cot Upper Platform:752	-->	CONFIG1

Configs Unused =
Worse performance

Manage Configurations

Assembly	Uses	Configuration
VET COT - ACTUATED:752-0039;	-->	MAX HEIGHT
Vet Cot Lower Frame:752-00		from VET COT - ACTUATE
Vet Cot Straight Legs:752-		from VET COT - ACTUATE
Vet Cot Bent Legs:752-0039		from VET COT - ACTUATE
Vet Cot Lock Bar:752-0039		from VET COT - ACTUATE
Vet Cot Upper Platform:752		from VET COT - ACTUATE

Configurations - 2



These buttons do not “solve” constraint problems. If an instance cannot be constrained in the desired fashion, reconsider the hierarchy structure. You may be requesting an impossible scenario. Avoid Unusing configurations unless the design requires it.

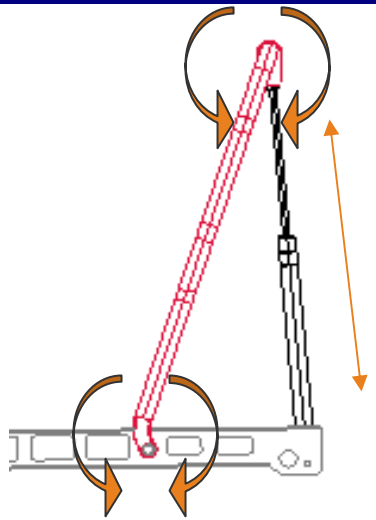
Configurations - 3

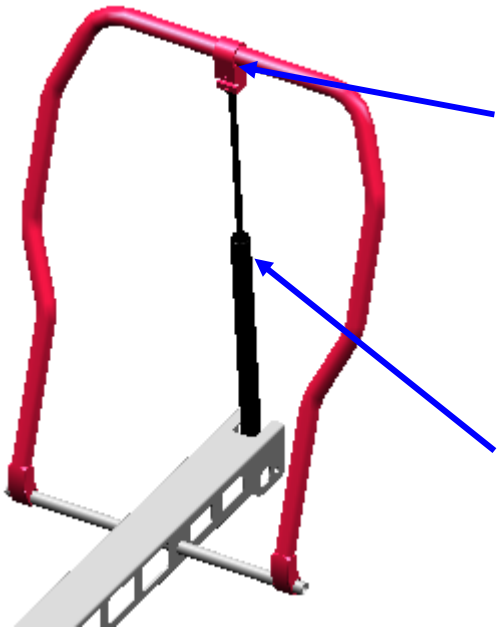
A subassembly whose configuration is unused is solved as if the subassembly wasn't there, and the pieces contained therein were assembled directly in the parent assembly. So, rather than solving one item (the subassembly position), I-DEAS must solve the position for every part in the subassembly.

In some situations, this can't be avoided. In those cases, it is particularly important that your subassembly is fully constrained. However, necessary motion will need to remain available. ⟶

Configurations - 4

Example: Backrest Assembly

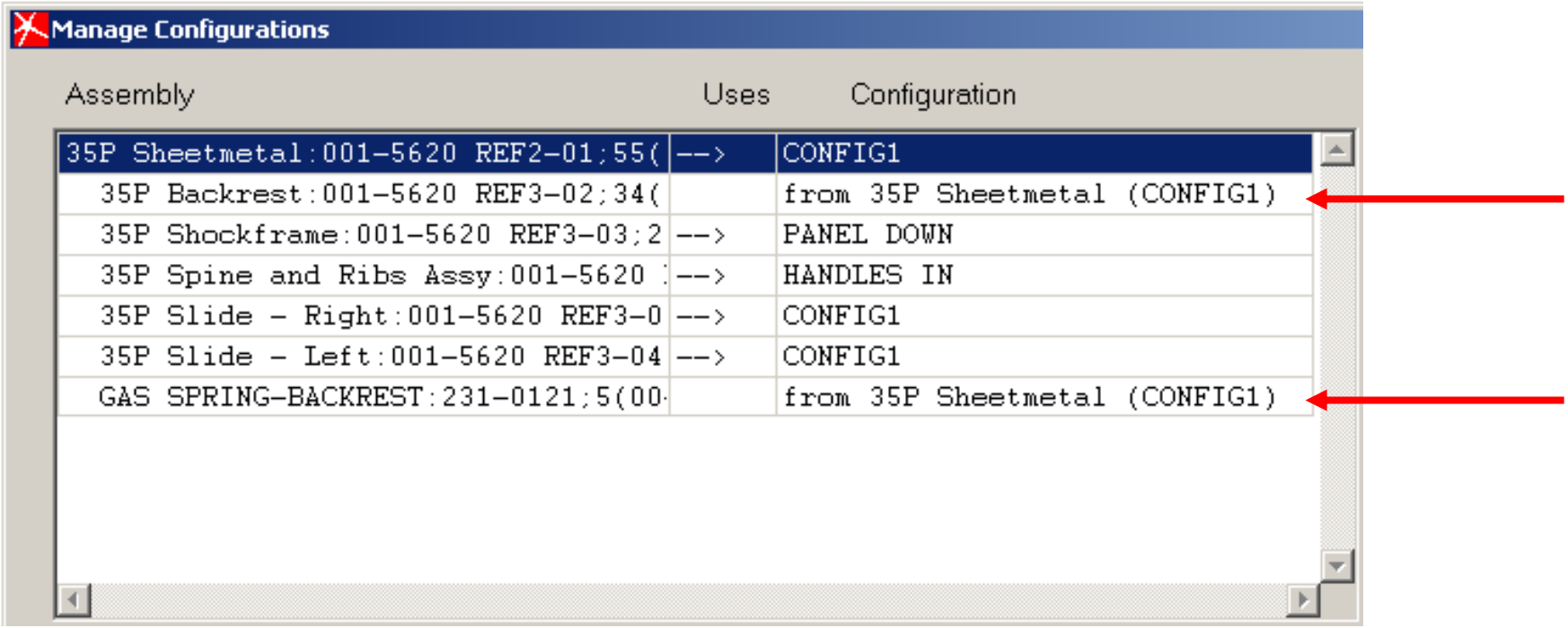




Extrusion must be on tube before tube is bent.
Documented as an assembly.
Extrusion must be free to rotate around tube.

Gas shock either must drive assembly, therefore requiring many configurations, or must be free to change length.

Configurations - 5



In this situation, unusing could not be avoided. In general, it is not desirable.

Summary

- Spread large assemblies across multiple model files.
- Break the hierarchy down into subassemblies that contain 50 or fewer instances.
- Use BORN parts at key attachment points.
- Fully constrain all instances.
- Avoid Unusing configurations as much as possible.

The End

Questions? Comments?

Brian Slick

bslick@ferno.com