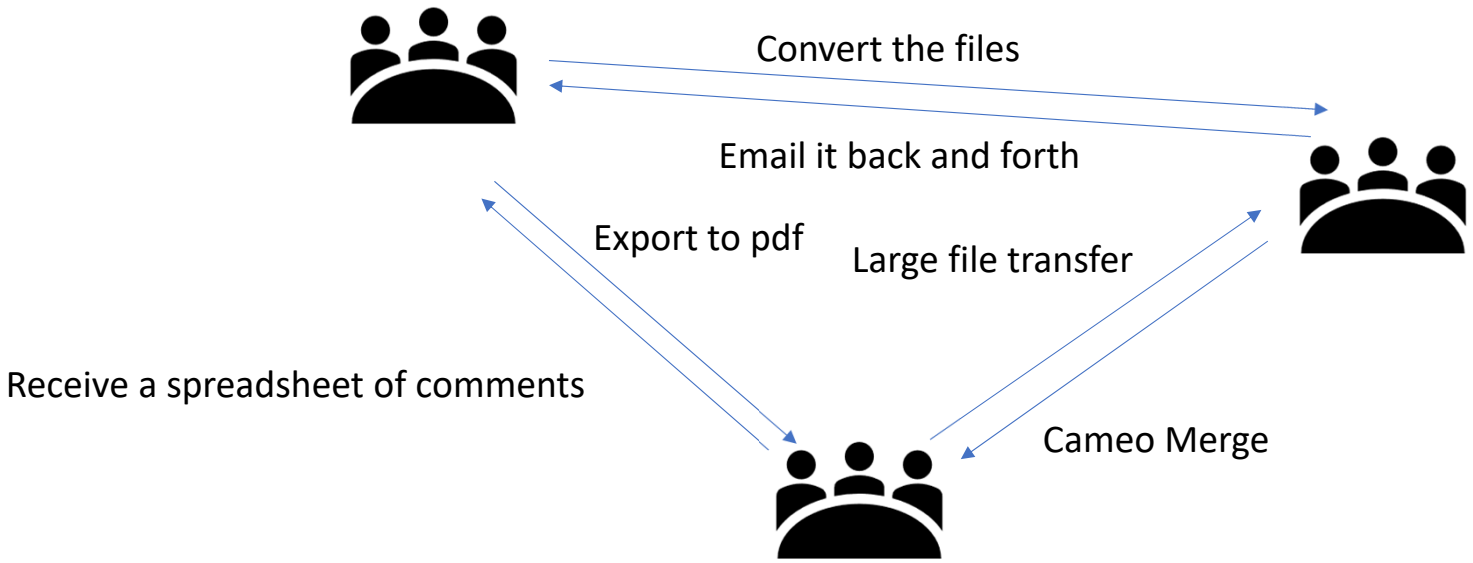
Approaches for Collaborative Modeling in Cross Company Environments

Amanda Weissman

Principal Systems Engineer

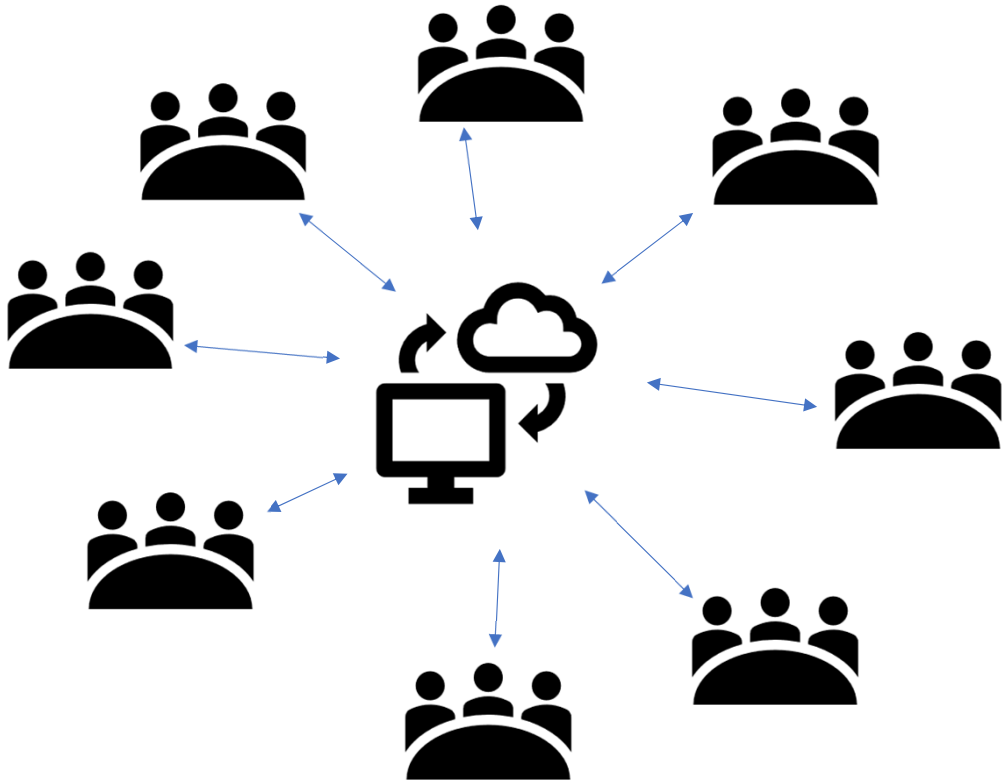
How do we share models today?



- Teams may be geographically co-located but are on different information systems
- Manually intensive process

Challenges: Speed, Efficiency, Morale

What do our teams of the future look like?



- Teams geographically dispersed and on different information systems

Goals: Speed, Seamless Sharing

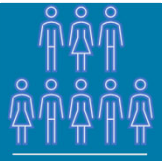
What are our Use Cases?

Systems Engineers

Concurrent design and documentation in the model

Safety Engineers

Provide input to systems engineering models and perform Safety Analysis



Stakeholders

Review and provide feedback

Developers

Review, provide feedback, ensure compliance to model, document SRVM results

Technical Hurdles

- 1 tool to rule them all: Rhapsody or Cameo
- Who provides Cameo or Rhapsody licenses?
- Are we all using the same Cameo or Rhapsody versions?
- How do we all access the same Teamwork Cloud server? Or are there alternatives?
- Not assumed to be hurdles:
  - Style Guides – make an agreement and stick to it

Common Tool with Common Version was our first hurdle

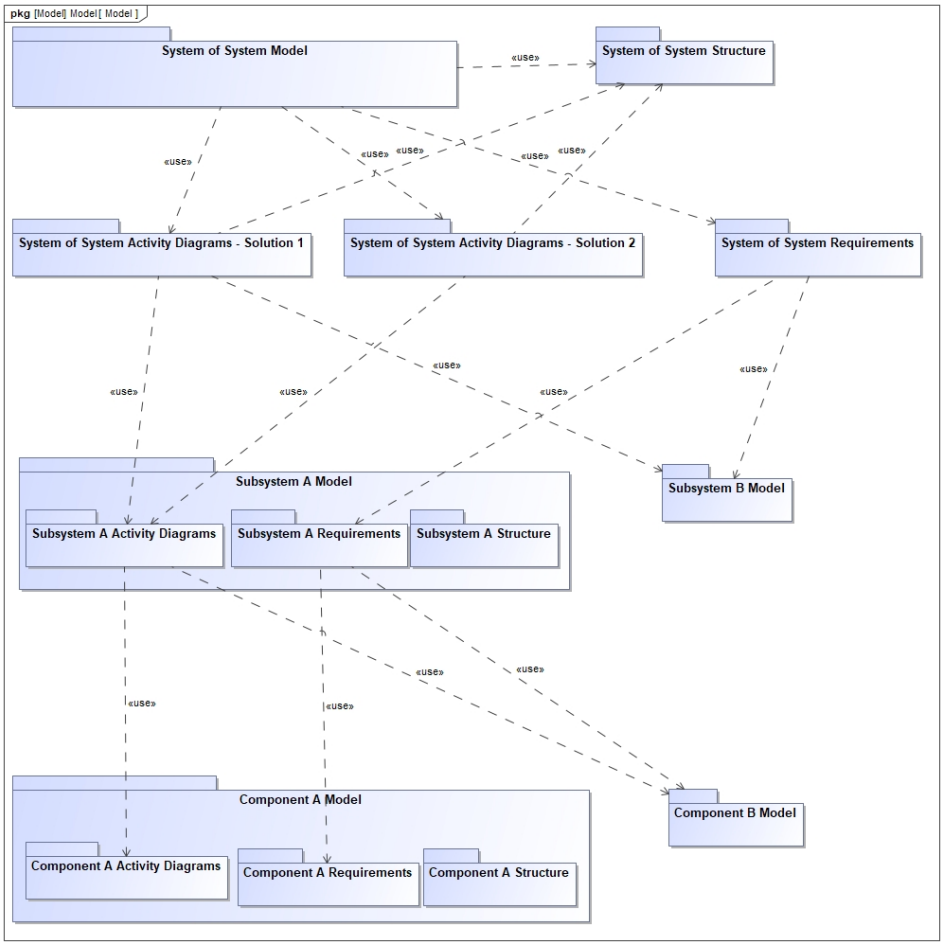
~~Technical Hurdles~~ Solutions

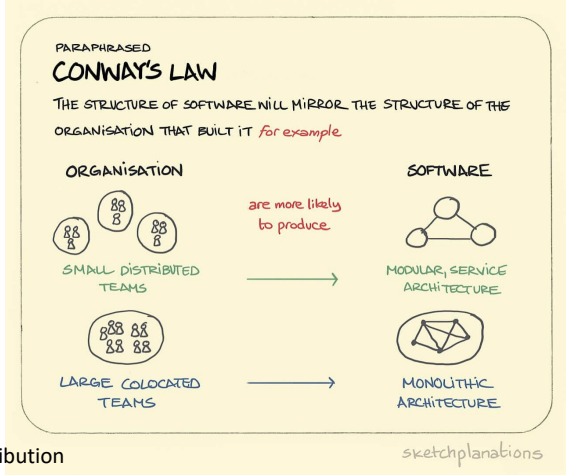
- 1 tool to rule them all: ~~Rhapsody or~~ Cameo
- Who hosts/pays for Cameo or Rhapsody licenses?
  - Company
  - Pool of licenses checked out through ecosystem and license server
- Are we all using the same Cameo or Rhapsody versions? Coordinated through Modeling Guild
- How do we all access the same Teamwork Cloud server? Or are there alternatives?
  - Git – available with CAC access

It's an imperfect solution, but we have 5+ companies, multiple government organizations, UARCs, and FFDRCs collaborating in real time!

Implications of the Solution

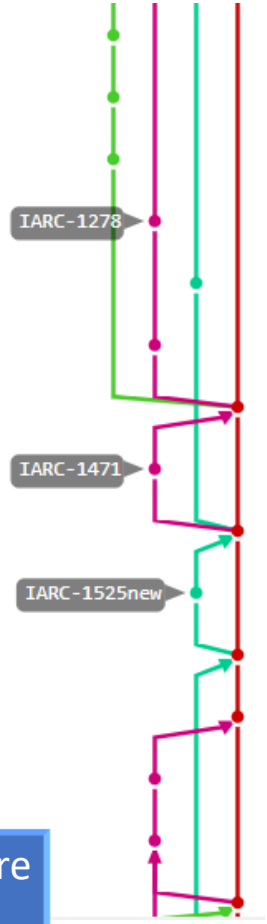
- Model structure
  - Cannot be one big file
  - Multiple model files (mdzip) aligned model structure to team structure which is aligned to architecture
  - 1 git repo per independently controlled model – ie JCSFL model, Top Level Requirements, etc.
  - Project Usages – need to be read only or managing mdzip's becomes extremely complicate





Implications of the Solution

- Learning curve
  - Developers familiar with git, everyone else needed to learn it
- Git limitations
  - Cannot work on the same mdzip at once because the merge is not automatic. Cameo Merge causes challenges. SysML 2.0 may resolve this issue.
    - Agile teams manage multiple team members in one file during daily standup
    - Cross team coordination for key files (such as System of System Structure Model) managed in Modeling Guild collaboration channel
    - Need to investigate file locking or other management strategy
  - Branching and merging strategy – branches must be short and merge often



Solution scales to complex system size if model structure aligns to team structure to mitigate Git limitations

Questions?