# Solving the Problems of Release Management

Center for Information Development Management

JoAnn Hackos, Director

# Today's presenters

- Dan Tong, Siemens
- Bill Gearhart, Comtech
- JoAnn Hackos, Comtech
- Chip Gettinger, SDL
- Members of the audience

# Change Happens!

Now – what are you going to do about it?

# Goals for today

- Explain to ourselves what we mean by release management
- Specify our need for “branch and merge” functionality from our content management vendors
- Explore alternative methods of managing content change

# The Problem

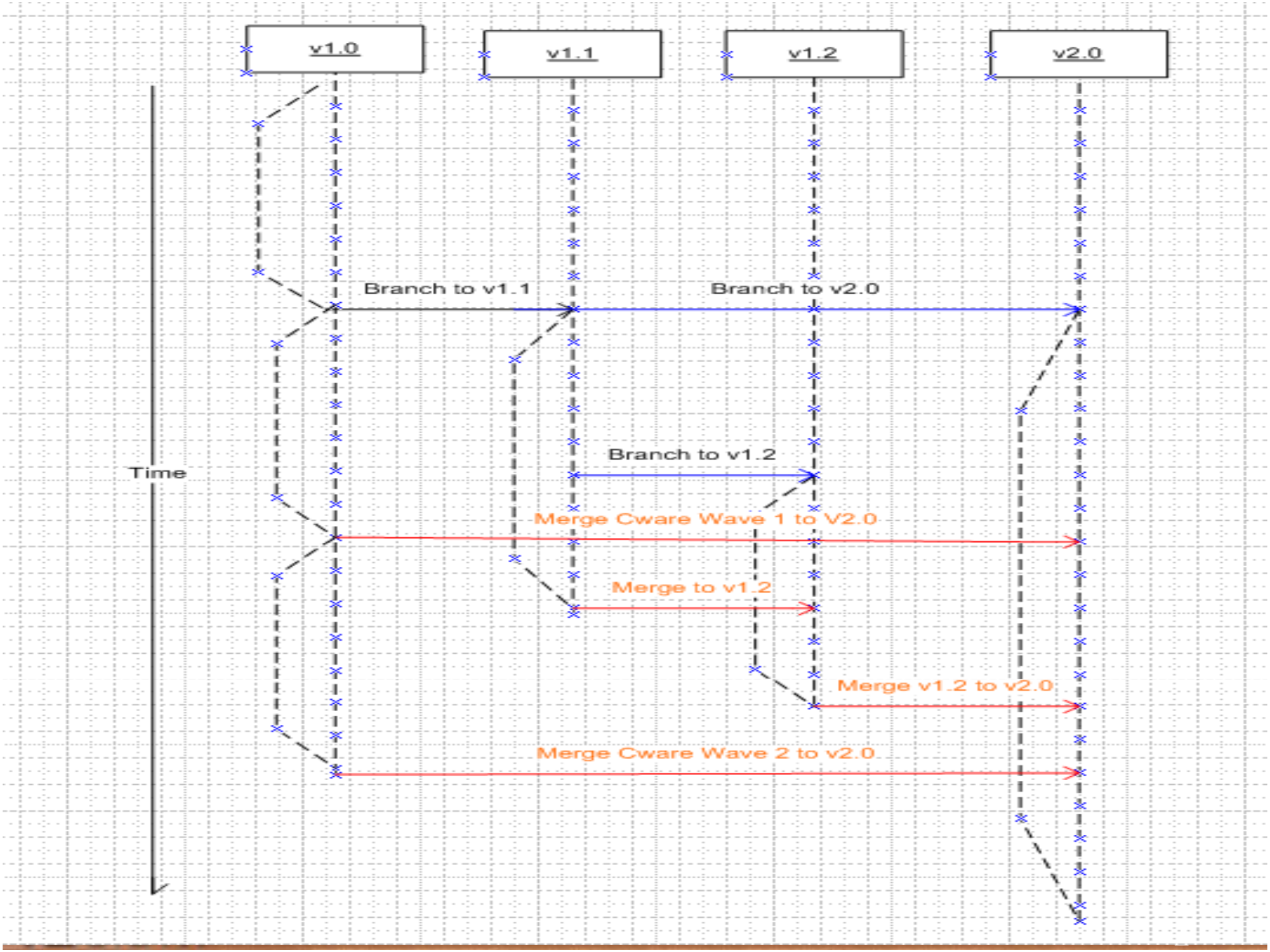
- Technical products are released on shorter and shorter schedules with multiple variations between releases
- Writers work on multiple releases at one time – making revisions to the same source files
- Changes made for one release must often be merged with other changes in other versions of the same topics

# Siemens scenario

- Learn what Siemens is dealing with in release managed between training and documentation

# Siemens scenarios

- The following scenarios describe the high level requirements for release management:
- Create and manage branches
- Concurrent editing
- Merging
- Labeling (archive)


v1.0
v1.1
v1.2
v2.0
Branch to v1.1
Branch to v2.0
Branch to v1.2
Time
Merge Cware Wave 1 to V2.0
Merge to v1.2
Merge v1.2 to v2.0
Merge Cware Wave 2 to v2.0

# Create and manage branches

- Project manager decides on the starting point and integrates those files into the new (v1.0) branch.  Writers check objects in and out of the branches.
- The project manager decides when the next branch is needed. Branches can be created at any point, for example:
  - The v1.0 tdoc release finishes and the project manager creates the v2.0 branch for that work to start.
  - The v1.0 courseware is still being worked on in the v1.0 branch.
  - At some point, the PM creates the v1.1 branch for a tdoc release.
  - At a later point, the PM creates the v1.2 branch for a tdoc release.

# Concurrent editing

- Work continues in all 4 branches and writers can concurrently edit the same object in any branch. Each branch contains a unique instance of every object. For example:
  - Courseware Chris checks out a topic 1234.xml in the v1 branch.
  - At the same time Tdoc Tracey checks out the same topic, 1234.xml in the v2 branch.
  - Courseware Chris edits the topic for the courseware release.
  - Tdoc Tracey edit the topic for the tdoc release.
  - Chris and Tracey check in the 1234.xml and release the deliverables as required.

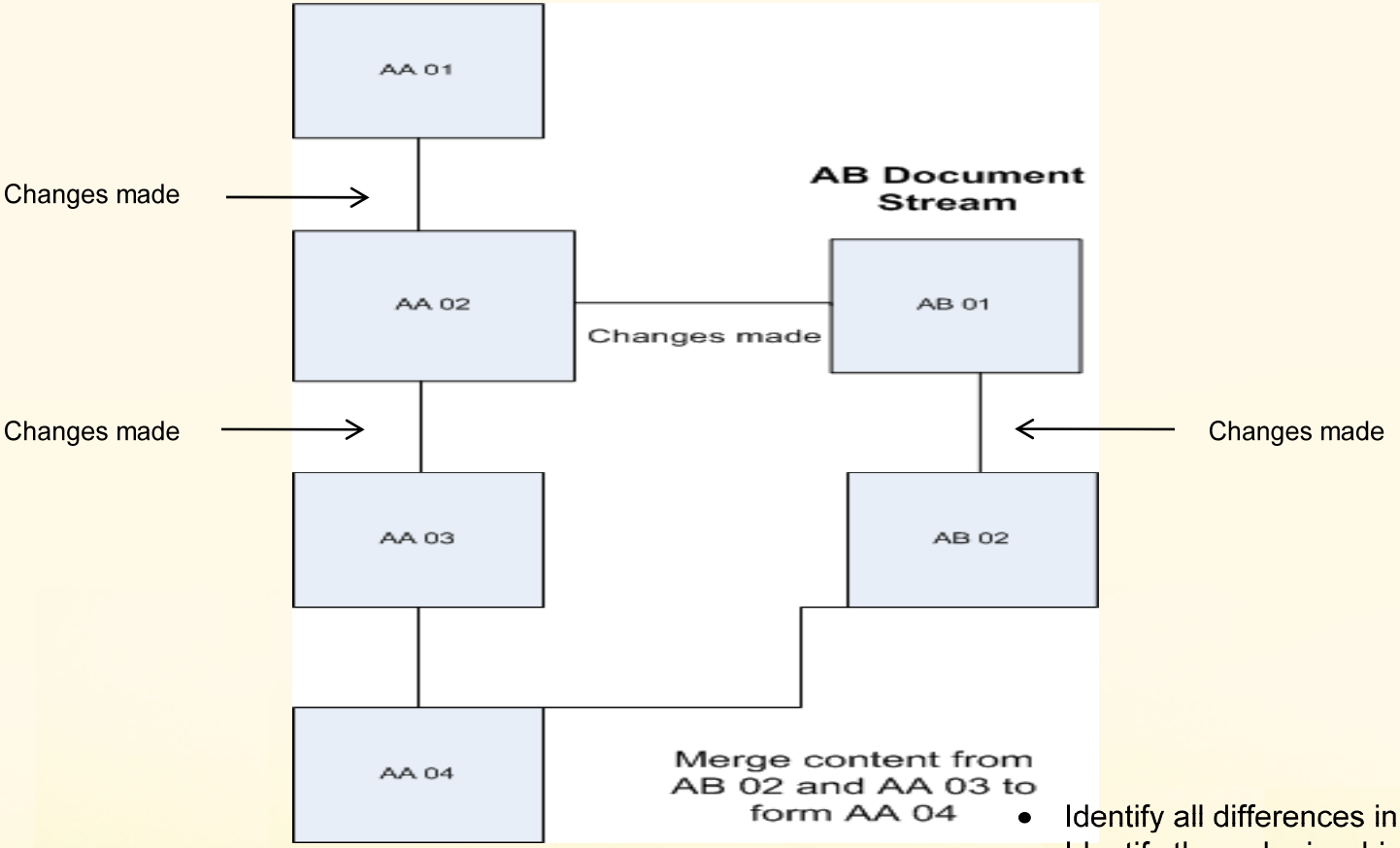
# Merge changes

- The project manager decides if and when merges need to take place between branches. For example:
  - When the v1.0 courseware Wave 1 release is finished, the project manager decides to merge those changes to the v2.0 branch so that the v2.0 courseware work can begin.
  - The project manager decides how to resolve all files scheduled for the merge. The resolve can be handled in two ways:
    - Auto-resolve: In most cases, the files will contain no conflicts and can be handled by an automatic resolve.
    - Interactive resolve: Files that contain conflicts (i.e., the same line of code has been changed) will be flagged as requiring an interactive resolve. The PM will need to contact the writers and make a decision on how to proceed.

# Label (archive) a branch

- The project manager decides when work in a branch has reached a significant milestone and should be labeled.
- The branch is labeled accordingly, for example, v1.0_tdoc_final.
- Labels can be accessed later to create branches for re-release of content.

# Comtech Scenario




- Identify all differences in the draft AA04
- Identify the order in which all changes were made (date, time, responsible party)
- Assist the writer in making decisions to accept, reject, or revise before incorporating

# Multiple release teams

V 1.0 release

Release team 1.0

9 months

V 1.1 release

Release team 1.1

9 months

V 1.2 release

Release team 1.2

9 months

V 2.0 release

Release team 2.0

9 months

Key Management Question

"Do we have the people?“
asked before
“What will it cost?”

- Smaller teams
- Better related to development teams
- More time for innovation
- More time for planning

# Comments

- Let’s hear from our participants
- What problems are you dealing with
- How are you handling them today
- What would you real like to happen