Cracking open DOORS… and taking a peek

Anne Heavey

For LBNE

Jan. 20, 2011

Important DOORS Features

- Versatility (can handle more than just requirements, flexible structure)
- Directional links provide traceback (or flow-down) capability
- Modular structure (can split up into different areas, but they can refer to each other within the Project)
- Hierarchical object organization within a module (e.g., levels 1, 1.1, 1.1.1, etc.)
- Configurable attributes (metadata)
- Configurable views and reporting (Getting PDFs requires export to Word, then hand-formatting)
- Configurable access rights by module and by hierarchy within module
- Import/export capability from/to MS products (e.g., Word, Excel (tested)), html and others
- Automatic history, versioning, create/modify dates and so on
- Can create custom data-input forms to facilitate data entry
- OLE (object linking and embedding) support
- Can propose changes to objects, track them (I haven't done this)
- Has a component that serves it all on the Web (FNAL didn't buy this)

DOORS Limitations

- Doesn't run on Mac; access it through terminal server (but there's a server install at FNAL that's backed up)
- Expensive per-user licensing
- For info shared by different modules: need to put in tables; can input tables by reference, but not other modules.
- (from Steve Acheson at DUSEL; has lots of DOORS experience)
  - Some bugs, mostly corrected now
  - Does configuration management by object very well (traceability and export), but not a great database
  - Does not make reports so well (add in cover page, TOC, etc. as OLE; then export to Word, then format)
  - "Painful" for one person to import everybody's Excel spreadsheets
  - He suggests also looking into Topcased – freeware run by Airbus and other companies. Made for modeling system engineering.

What is TOPCASED?

TOPCASED is an integrated System/Software engineering toolkit compliant with the requirements of critical and embedded applications. It covers the stages from requirements analysis to implementation, as well as some transversal activities like anomaly management, version control, and requirements traceability.

TOPCASED is strongly model-oriented : not only TOPCASED provides model editors, model checkers and model transformations, but is also itself based on modelling and code generation.

TOPCASED is a meta-tool : you can develop your own graphical editors and model transformations using TOPCASED.

TOPCASED is available under an EPL (Eclipse Public licence).

(from their documentation)

TOPCASE claims these functions

- Addresses requirement management, fine-grained model coverage Import requirements from document (word, excel) coming from Doors, ReqPro…
- Cover those requirements by current requirements attached to model elements
- Check upper/current traceability through specific view

A word about DUSEL from Steve A…

Regardless of the requirements management system LBNE uses, DUSEL will want the following information from us for each requirement that it, as the facility, will need to meet:

- Clear statement of each requirement
- Unique number for each, for tracking
- Verification method (test, demo, inspection, analysis) so that they can verify their compliance in a mutually agreed-on way.

Preparation for using DOORS

- Lots of good doc on
  - How to state a requirement properly
  - How to use DOORS to manage requirements
- Many features; requires prep to use effectively
- To use basic/mid-level features, it seems not too complicated
- Determine set of documents needed at the end (next slide shows example, based on DUSEL's)
- Determine modularity, categories, attributes, and so on
- User recommendations
  - Erik Gottshalk says to bring in DOORS consultant to help us configure our structure, hierarchy, modularity, etc.
  - His project did that; was very helpful
  - Steve Acheson at DUSEL stresses importance of preparation; determine the final doc set that you want, consider audience of each and have reqs 50% set in stone before starting input to DOORS.

Document structure sample

- Level 1
  - Project Requirements Document
    - Overall Project Goals and Requirements of LBNE
    - Audience: funding agencies
  - Subproject Requirements and/or Commissioning Documentation (1 per subproject = 4 docs)
    - Overall subproject goals and requirements
    - Audience: funding agencies, collaboration
    - Interface requirements on facilities
    - Audience: FNAL, DUSEL

Doc structure cont'd

- Level 2
  - Overall reqs/params for each subsystem of subproject (e.g., for LAr: Cryo, TPC, DAQ, etc.)
  - Audience: collaboration, subproject, contractors, facility
- Level 3
  - Requirements for each major component of subsystem (e.g., for Lar TPC: APA, CPA)
  - Audience: subproject, contractors

Doc structure cont'd (ii)

- Integrated Subproject Interface Requirements Document
  - Requirements that each subproject will be expected to meet when integrated into experiment and eventually commissioned at the facilities.
  - Audience: facilities, project, collaboration
  - (These requirements are traced to higher-level requirements that govern both facility and experiment subprojects.)
- Contractor Requirements Documents
  - Requirements against which each contractor will be verified during commissioning.
  - Audience: contractors

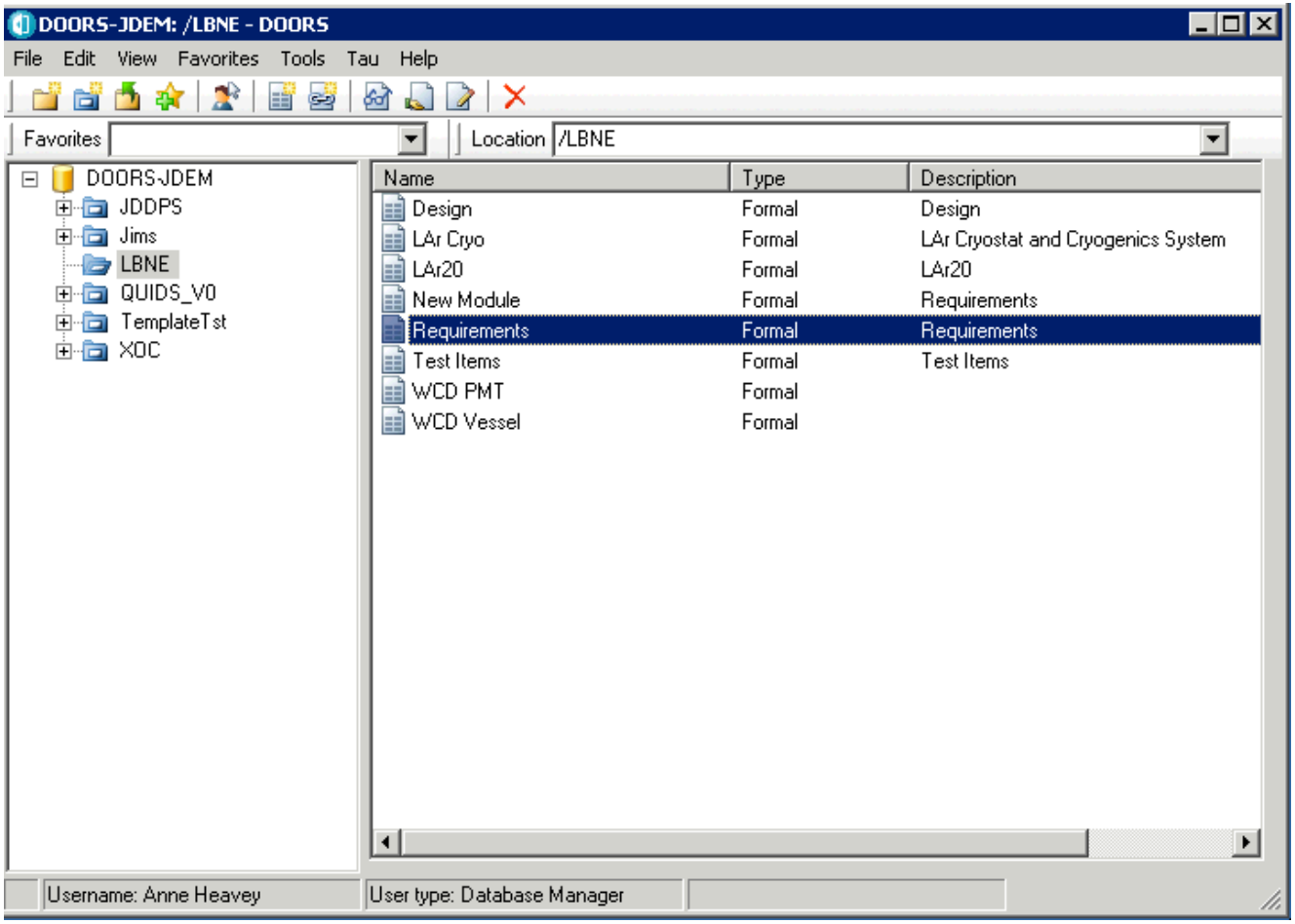
DOORS Usage at Fermilab

- FNAL has it installed on a Windows server, has 2 licenses (I'm borrowing one)
- On Mac, I use "terminal server" app to connect to Windows interface, open DOORS from there
- I created new "project" named "LBNE"
- Under that, several "formal modules" (sort of like separate files)

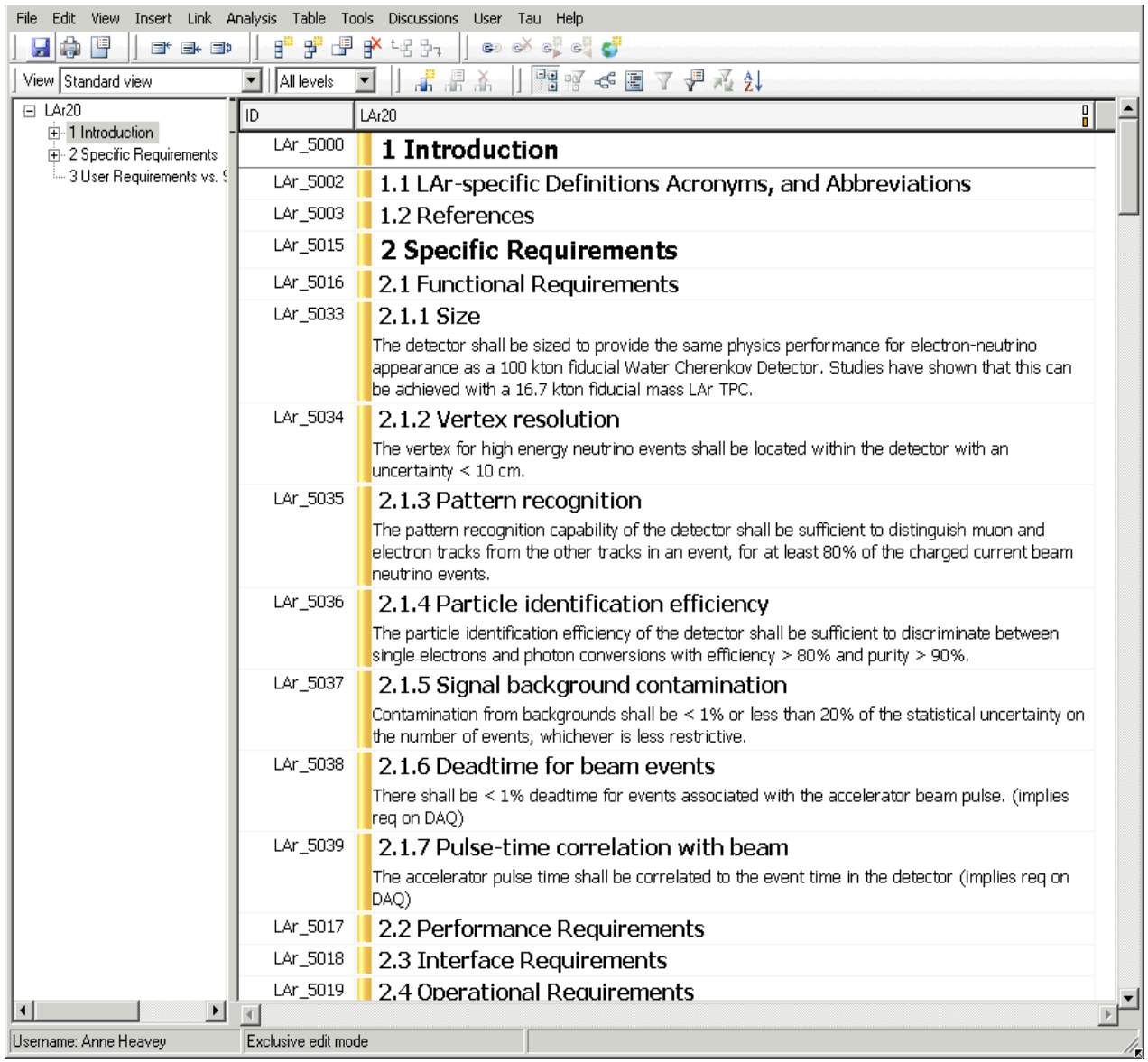
Modules

- One level of module (i.e., not hierarchical) but “sister” modules are “connectable” via links
- Modularity flexible; I’m playing with the idea of using one module per WBS level 3 (i.e., one per chapter in CDR)
- Within a module, hierarchical structure is available for contents
- Module templates are available
- In on-screen window, looks like Word doc in outline mode
- Items auto-numbered; unique identifier; can take module-wide prefix

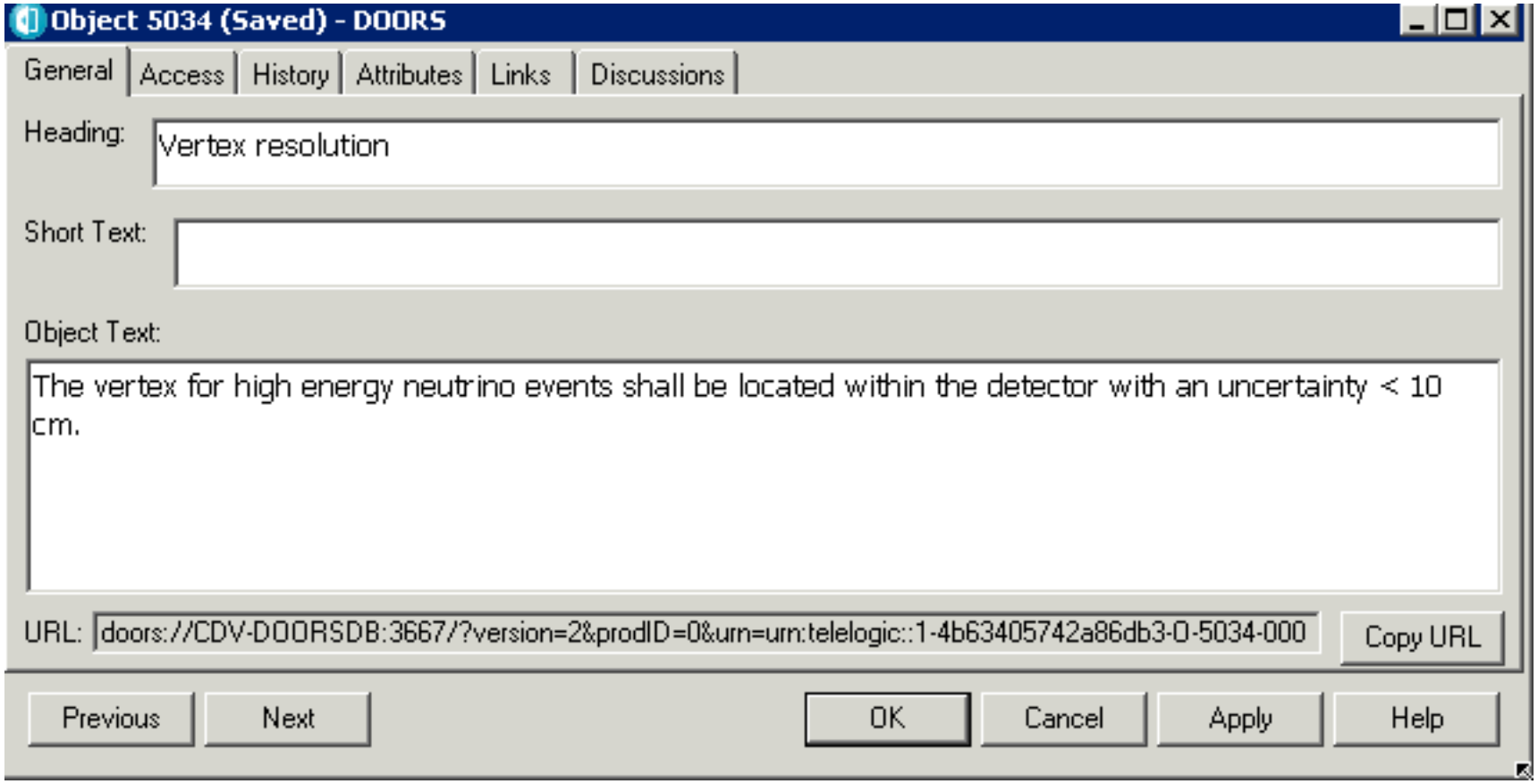
Sample module listing for a project



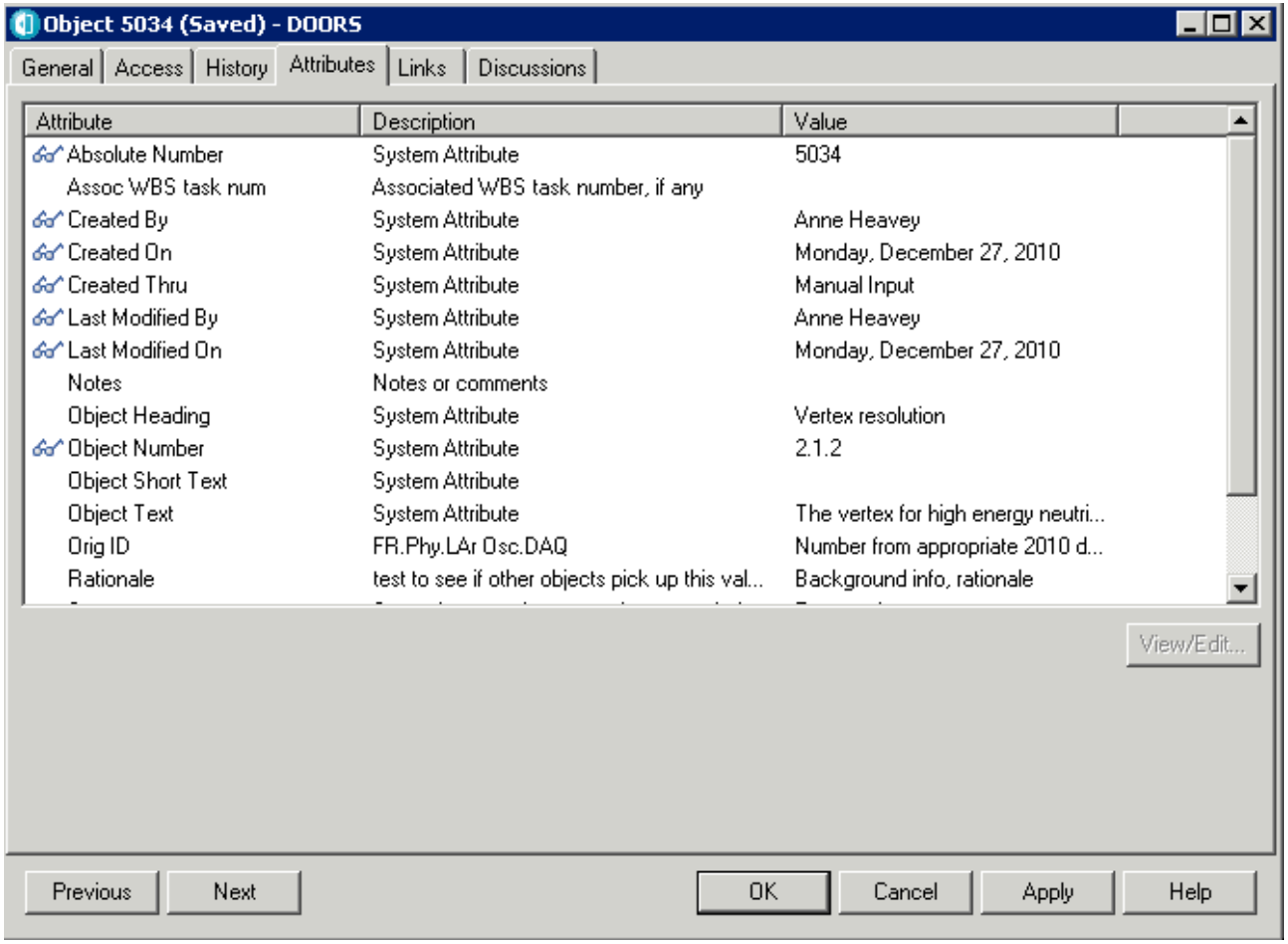
Sample module screen



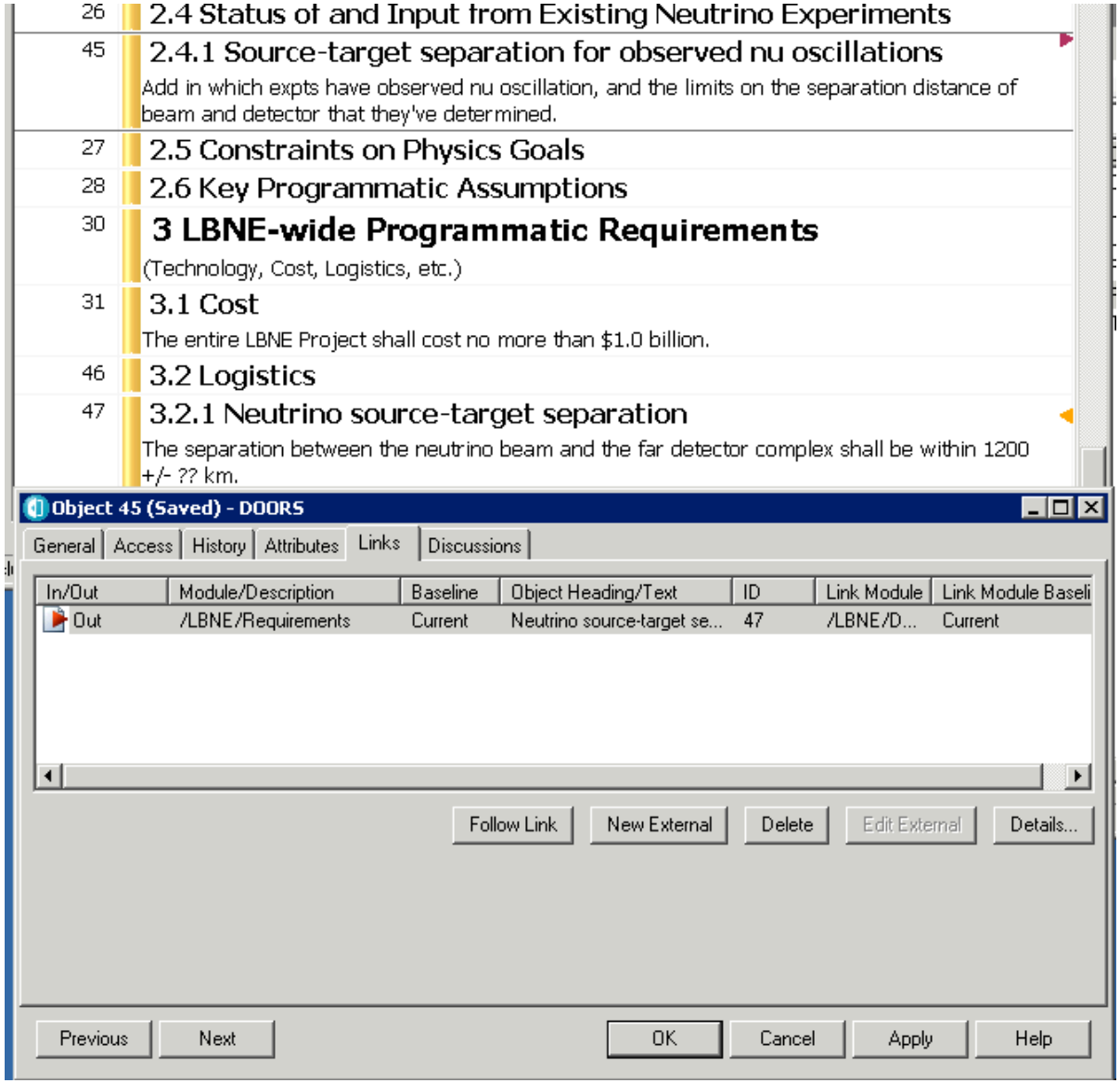
Sample *Properties* pop-up



Sample *Attributes* list



Sample Flow-down link



The next few slides are Anne's interpretation of what makes a good requirement and a good structure for documenting them based on both DOORS doc and other sources.

Prep for writing good requirements

Before writing requirements, identify the system's:

- needs
- goals/objectives
- constraints
- missions
- operations concept
- budget
- schedule
- management/organization

Requirement characteristics

Each requirement must be:

- necessary, verifiable and attainable
- clearly, unambiguously written
- about one single issue (i.e., avoid "and”)
- about what the 'system' needs to accomplish, not what a 'component' of it should do
- NOT about implementation (the HOW), but rather about WHAT is needed. (test: ask yourself why the req is needed; if this leads to another requirement, it's probably an implementation statement)
- NOT about operations (don't think about HOW you'll do something with the system, think of what the design needs to make possible for your tasks, and let the designer figure out how to design it)
- specified to the appropriate level, considering function, cost, schedule, etc. Over-specification leads to $25k coffee pots (e.g., if triple redundancy is overkill, don’t specify it)
- appropriately stringent (e.g., don't say "exactly 100 ft" if it can be 100 +/- 3 ft)

Requirement terminology

Terms (standard usage in govt and industry, according to author):

- Requirements use SHALL
- Statements of fact use WILL
- Goals use SHOULD
- Other terminology guidelines:
  Requirements do NOT use: “is”, “was” and “must”
- Requirements do NOT contain unverifiable words like “maximize”, “user-friendly”, “adequate”, and so on
- NEVER use: “but not limited to”, “etc.”, “and/or”

Draft attribute list

- System-generated (the principal ones)
  - Number (with optional prefix)
  - Accepted (y/n)
  - Creation/modify dates and by whom
  - Title and text
- Additional ones proposed
  - Type (e.g., physics, programmatic, functional, performance)*
  - Rationale
  - Associated WBS number
  - Verification method
  - Verified? (y/n)
  - Notes

*Or if we use DOORS for more than just reqs, the types could be: Objectives, Assumptions, Constraints, Requirements, Design Parameters, and so on. Then requirements could have a subtype, if necessary.

Potential implications for CDR

- To minimize duplication, leave requirements out of text of CDR; possibly import into CDR; or, just reference appropriate requirements doc in each chapter.
- Use the requirements-to-design-parameter coupling in requirements doc as basis of description/discussion of each element in CDR.

Other CDR changes from 12/10/10 meeting

- Things to remove from individual volumes and collect into separate documents:
  - QA plan/process (but leave specific QA steps for each element in individual chapters)
  - VE material: goes only in Mike's VE database
  - Risks: go into Mike's Risk Register
- Has Project Office agreed to these changes?

Thank you.

- Discussion?