Architecture Breakout Session

J. Borbinha, D. Castelli, N. Fuhr, M. Mahey, P. Pagano,
A. Powell, H. Schuldt, C. Thanos, T. van Veen

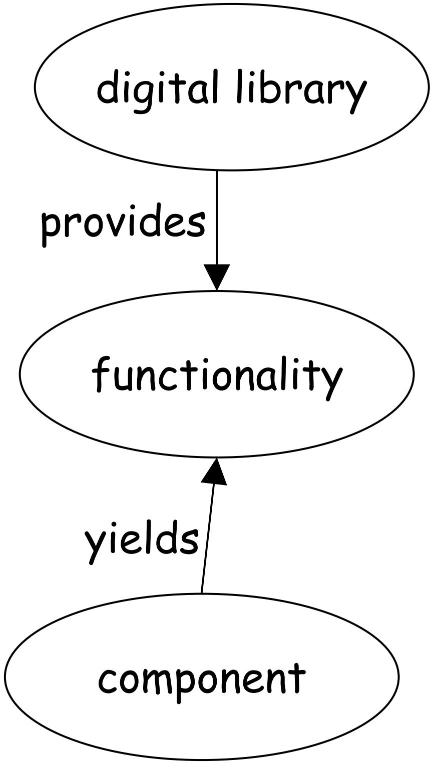
Top-down vs. Bottom-up …

- The Reference Model follows a bottom-up approach
  - Stating with users and user functionality, identifying components, etc …
- Top-down considers processes and end user requirements
  - Identify relationships between components
    - How do they interact, SLAs, …
  - Validate the components that have been identified / define them in an abstract way
  - Combine both approaches

… Top-down vs. Bottom-up

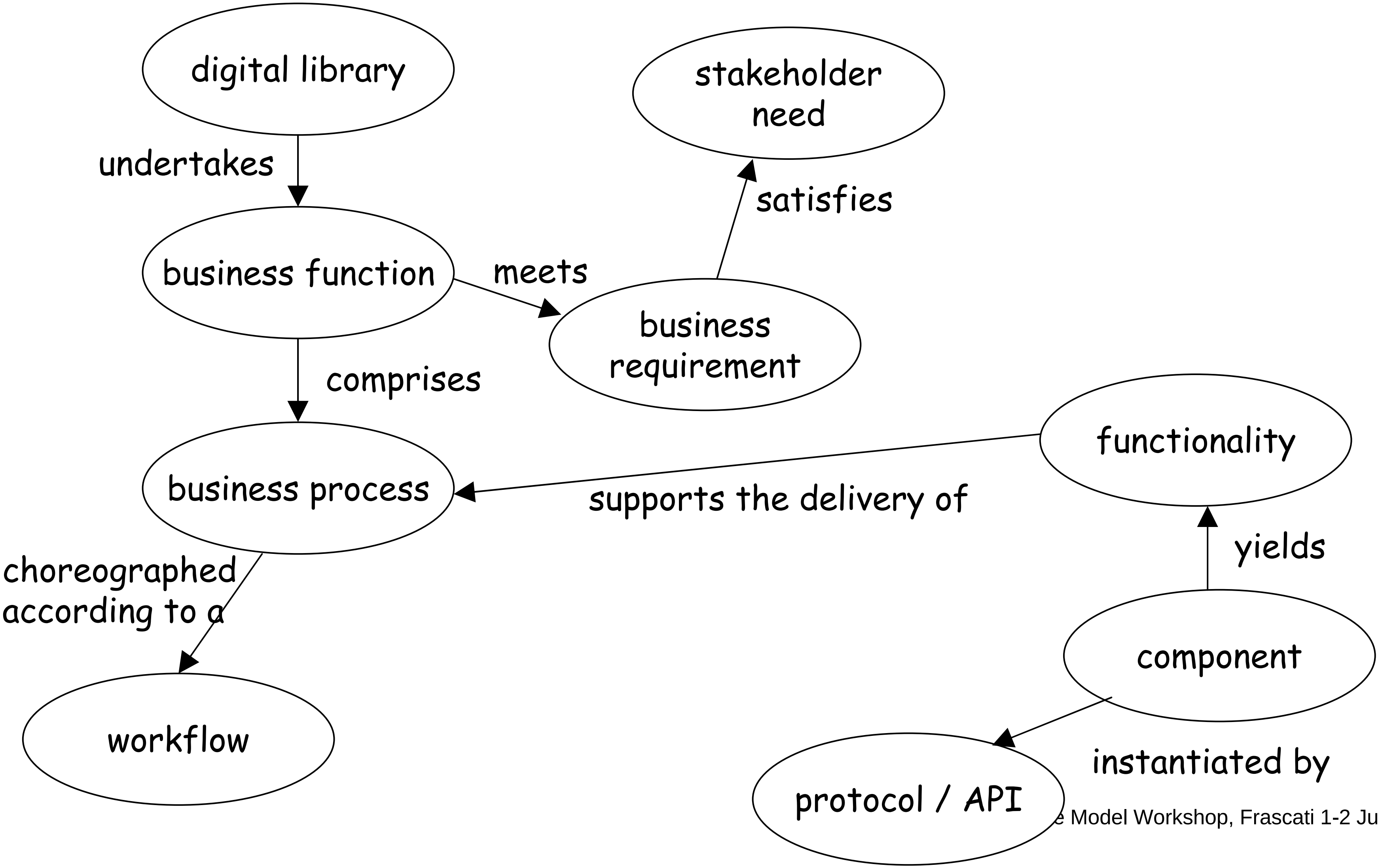
- Exploit for a methodology to derive a DL Architecture from the Reference Architecture based on user requirements

Current concept maps to simplistic?



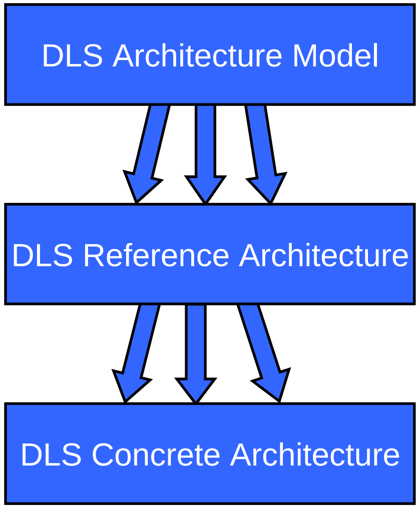
- current concept maps have something like this (e.g. fig 11, page 34)…
- too simplistic?
- ‘functionality’ provided by the DL is not that directly yielded by the ‘component’
- no ‘business requirement’, user-needs or process analysis from which to derive ‘components’
- therefore difficult to justify choice of ‘components’ in reference architecture design
- e.g. why does reference architecture design contain an ‘OAI harvester’ component?

Possible alternative concept map?



digital library
undertakes
business function
meets
business requirement
satisfies
stakeholder need
comprises
business process
supports the delivery of
functionality
yields
component
instantiated by
protocol / API
choreographed according to a
workflow

DelosDLMS Experience



- Clarify the relationships between the different levels (reference implementation / reference model)
- Use description of DelosDLMS (in another document?) to explain the Reference Architecture and to give an example on how to work with it

Other comments

- section 5 (reference architecture) doesn’t sit comfortably with the rest of the document
- linkages with rest of document not clear enough?
- in particular not clear what is ‘example implementation’, what is ‘reference’?
- gets concrete very quickly
- need to define components in abstract way – then say that these ‘abstract components’ can be instantiated using particular protocols and/or APIs