CMDB Patterns

Designing CMDB data models with good utility and limited complexity

Michael Brenner; Markus Gillmeister

Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities

Garching n. Munich, Germany

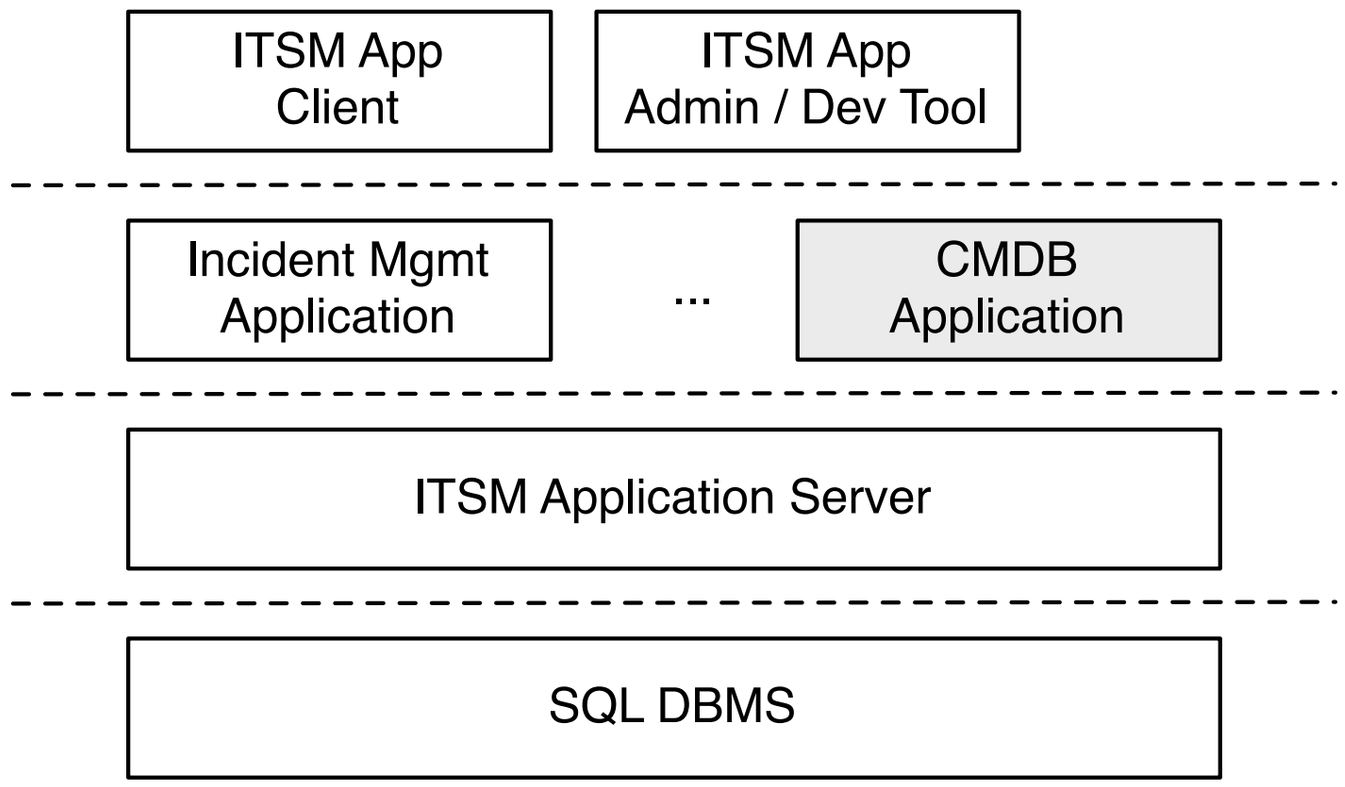
CMDB as a concept and as a tool

CMDB as a concept of ISO/IEC 20000 and ITIL

- *Configuration management database (CMDB)*
  Data store used to record attributes of configuration items, and the relationships between configuration items, throughout their lifecycle
- *Configuration item (CI)*
  Element that needs to be controlled in order to deliver a service or services

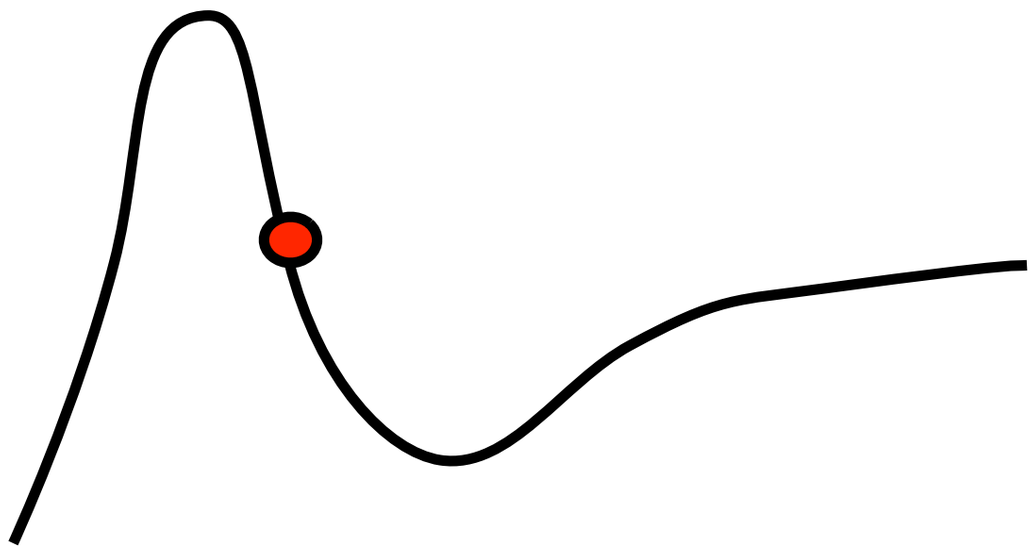
CMDB as a tool

- Usually part of a comprehensive ITSM suite
- Allows linking CIs to incident records, problem records, change records etc. and vice versa
- SQL basis usually quite noticeable (no true object orientation)



Status quo of the CMDB in practice

- About half of all CMDB projects fail [Gartner 2013], “Cause of death” is almost always complexity
- Part of the IT service management community questions whether implementing and maintaining a CMDB is justifiable from a business point of view:
  It is such an enormous undertaking that any organisation attempting it is going to burn money on an irresponsible scale.
  (Blog post by IT Skeptic: “ITIL’s dead elephant: CMDB can't be done”)
- CMDB is currently heading down into the “trough of disillusionment” in the Gartner hype cycle.



Position of the “Service View CMDB” in the hype cycle for IT operations management

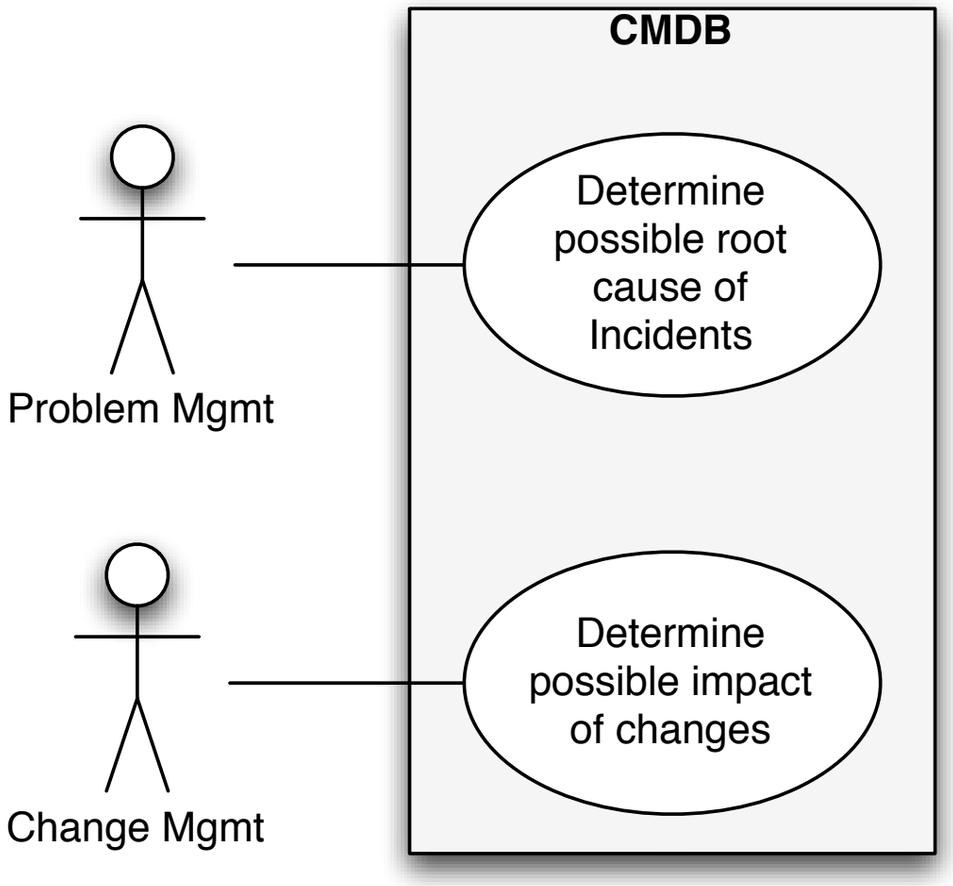
Towards a higher success rate for CMDB projects

- Status quo: Each organization designs its own CMDB information model. This is not likely to change in the near future!
  - Universally accepted common information and data model for CMDBs is nowhere in sight
  - Service providers' infrastructures, business models, company cultures etc. vary greatly, there might never be one CMDB information model to suit all needs
- What is needed:

Non-prescriptive, adaptable, pragmatic guidance on CMDB design

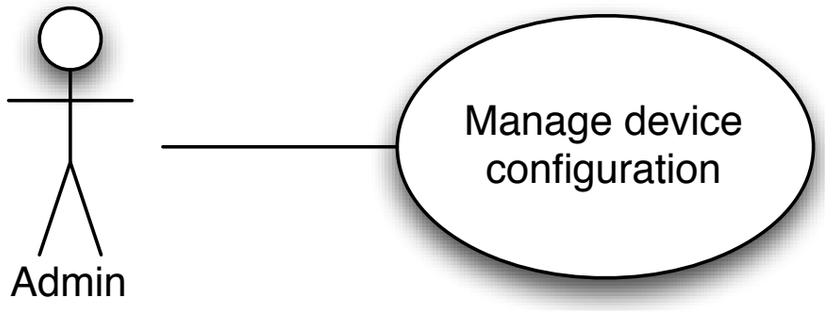
- Setting the right scope for a CMDB project
  -> clarification of CMDB requirements and prioritization of use cases
- Guidance on designing the information model
  -> CMDB patterns

Setting the scope: CMDB use cases



"Users at site A and site C have reported network connectivity issues.
What network infrastructure do these sites share?
Have there any changes been done to it recently?
..."

Software update to patch level 22 for all switches of type X4000 (sw1brz, sw2brz, sw3brz) requested.
What parts of the network infrastructure and which end user sites are affected?

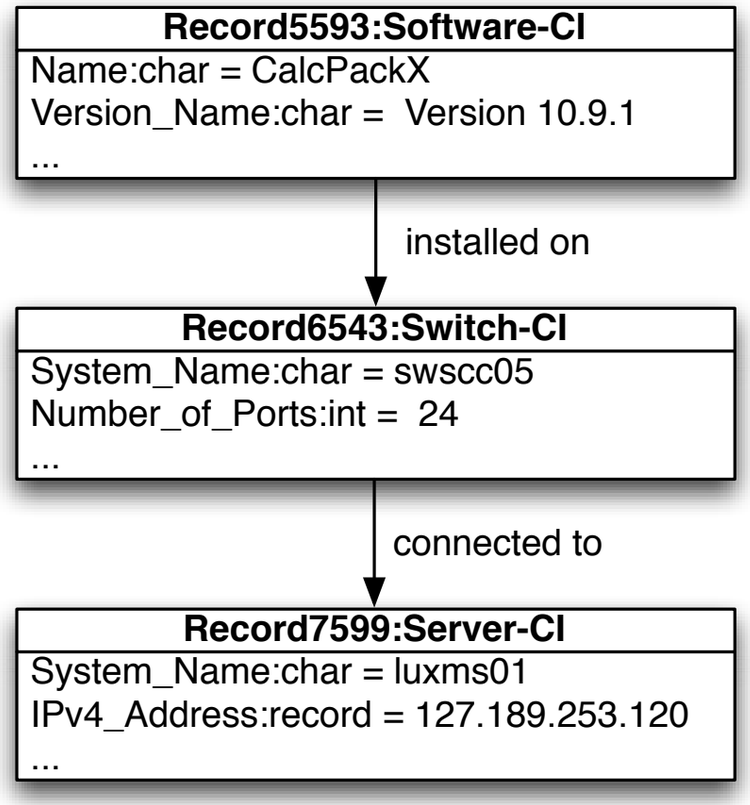


What is the value of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\PPP\EAP\88 on wsrv22.ads.example.com and srv5.ads-u2.example.com?

Goal: Benefit from finding the answers to these questions easier and quicker through use of the CMDB >> Cost of maintaining the CMDB

CMDB Patterns

- Majority of CMDBs are composed of
  - CI records, derived from a template and containing a number of attributes of a simple type
  - Binary and directed CI-relationships
- CMDBs become more complex, the more
  - attributes the records contain;
  - CI records there are;
  - CI-relationships there are per CI.
- CMDB patterns should
  - help to design CMDB information models that offer a good utility/maintenance ratio;
  - allow CMDB designers to share and discuss ideas using a common terminology;
  - limit the complexity of the resulting CMDB.

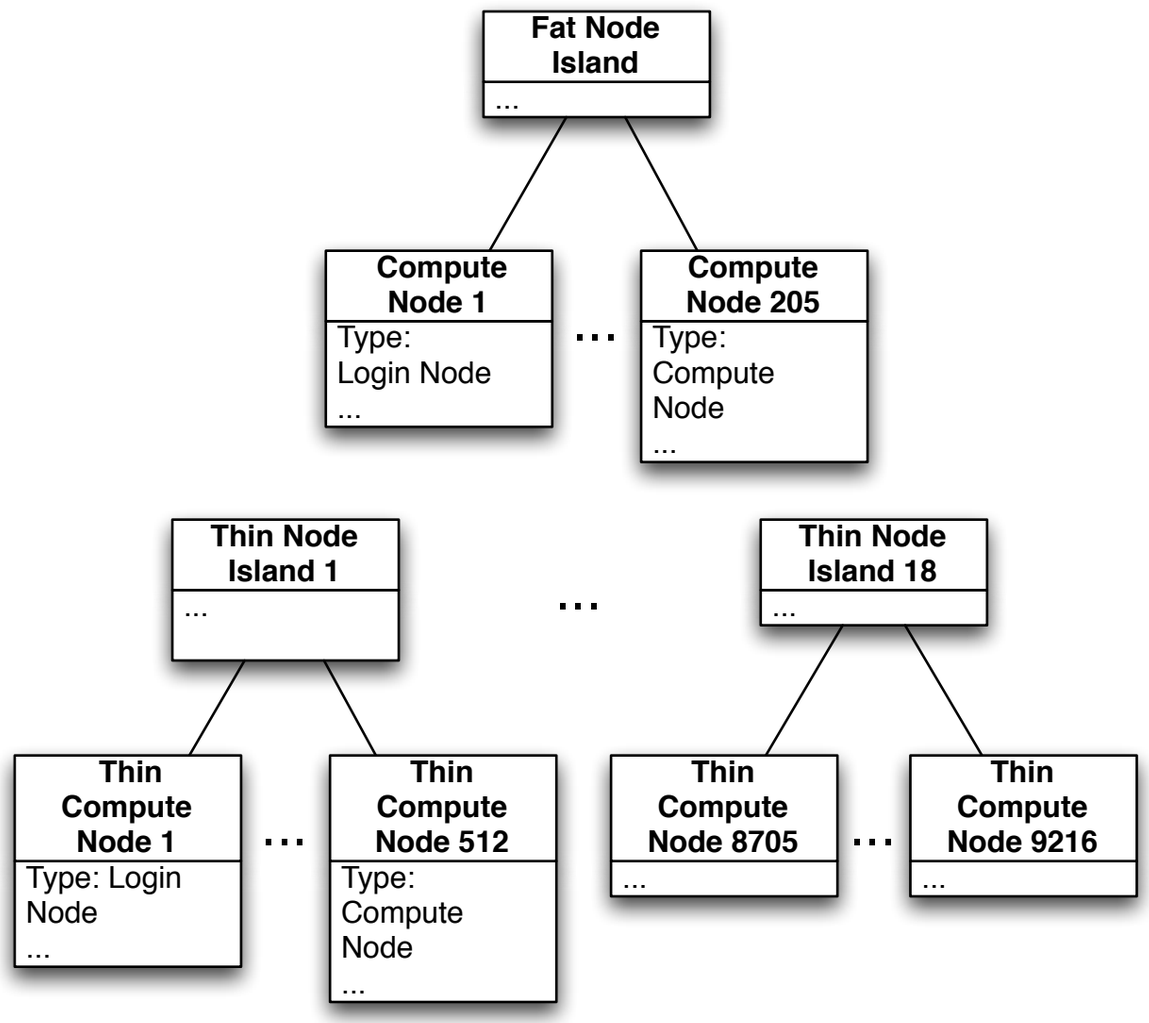


Pattern *Collective CI*

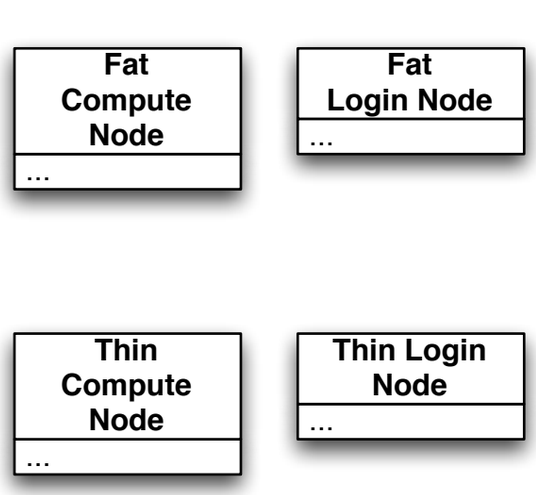
- *Description*
  Collective CI – Use one CI as a placeholder for many components
- *Leveraged circumstance (prerequisite)*
  There is an enforced policy to keep subsets of components in a standardized configuration.
- *Advantage*
  Number of CIs drastically reduced, important relationships easier to analyse, updating the CMDB less complex
- *Disadvantage*
  Individual relationships for each component not documented
- *Variations / Comments*
  Individual CI for each component, but use collective abstract CI for standard configuration groups: No reduction in number of CIs, but updating configuration easier. Individual CI-relationships are kept documented.

Collective CI – Example: Supercomputer nodes

Common approach:
1 CI per addressable node
+ 1 CI per island
>9000 CIs



Use of *Collective CI* Pattern
4 CIs



Pattern *Rich CI relations*

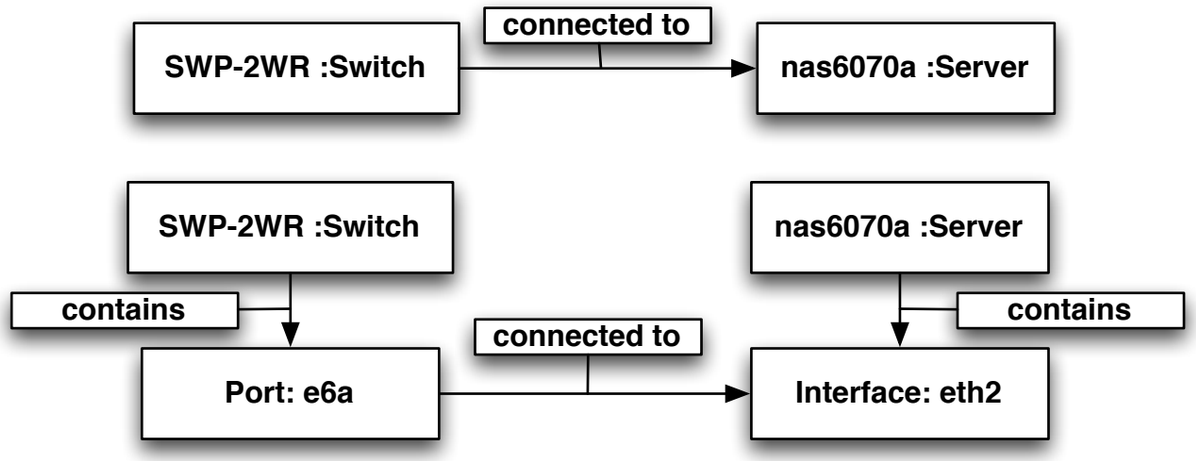
- *Description*
  Rich CI relations – Add editable attributes to CI-relations; preferably including *multi-value attributes*

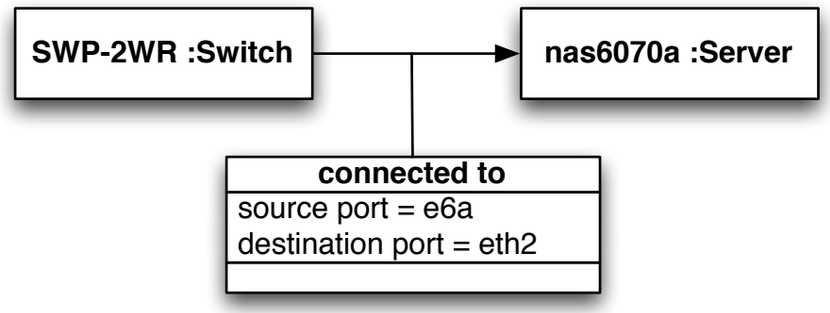
- *Leveraged circumstance (prerequisite)*
  Data model allows attributes to be added to relationships (usually given for CMDB based on relational databases)

- *Advantage*
  Number of CIs drastically reduced, major relationships clearer

- *Possible Disadvantage*
  Individual network components (NICs etc.) no longer controlled as individual CIs

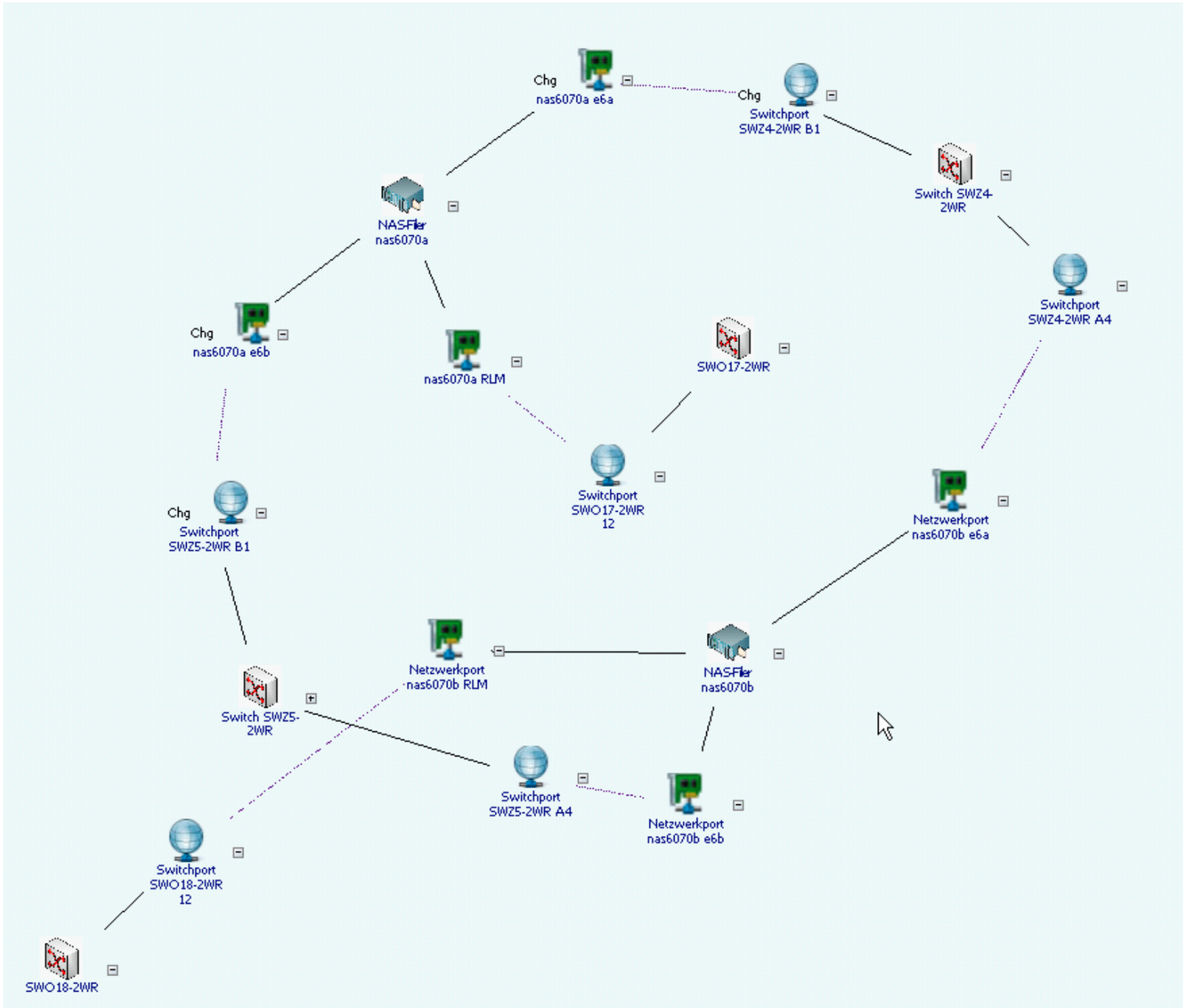
- *Variation / Comments*
  Pattern also useable for other documentation cases (e.g. mounting volumes)



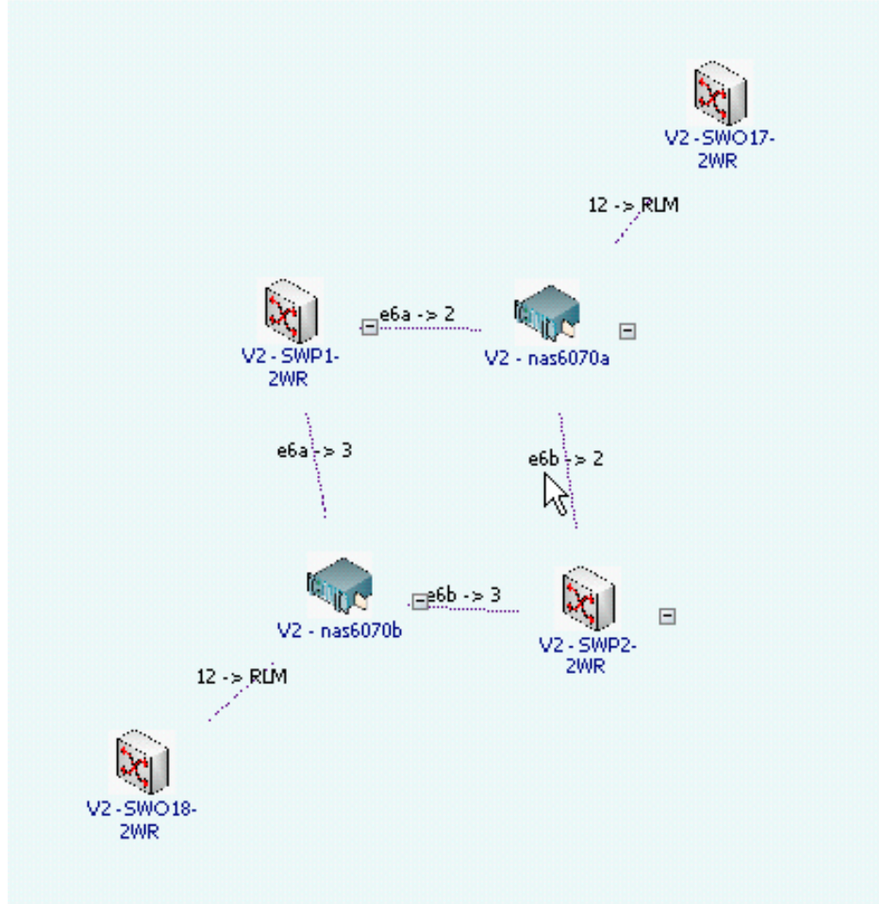


Rich CI relations - Example

Model 1



Model 2



Pattern *Multi-value attributes*

- *Description*
  Multi-value attributes – Enable record-type attributes in CI records
- *Leveraged circumstance (prerequisite)*
  Data model supports an attribute record type or is adaptable to suit one
  (e.g. support for comma-separated lists in attributes)
- *Advantage*
  Modeling of associations between system sub-components – e.g. how are IP addresses, MAC addresses and network interfaces bound to each other – much more efficient than with multiple-CI solutions.
- *Possible Disadvantage*
  Depending on the data model and implementation of the new type, some queries become more complex (e.g.”Which IP addresses are currently unused?”)
- *Variation / Comments*
  Multi-value attributes are ideal for modeling the network configuration of systems, but are also useful for describing mass storage components.
  Multi-value attributes and *Rich CI relations* are distinct patterns, but serve the same purpose – reducing the number of CIs with no or little loss of information – and are best used in combination.

Multi-value attributes - Example

Approach 1:

The way most out-of-the box example models look like. Common queries remain simple, but associations are lost.

Server	
Sys-Name	luxms01
MAC	00:07:E9:23:CD:59
MAC	00:07:E9:23:CD:58
IPv4	127.189.253.120/24
IPv4	127.189.253.108/24
IPv4	127.189.253.111/24
IPv4	127.189.253.102/24
IPv4	127.189.253.100/24
IPv4	11.77.6.1/24
IPv4	11.77.6.51/24

Approach 1 (CIM-like):

Conceptually clean, but many CIs and relationships. Many common queries (“what server has IP addresses 127.189.253.120?”) and consistency checks become difficult.

Approach 3 (multi-value attributes):

Associations are kept, common queries remain simple, while some others (“which IP-addresses are free?”) become more complex and need to be predefined.

Server			
luxms01			
eth0	00:07:E9:23:CD:59	127.189.253.120/24	luxms01.example.com
eth0	00:07:E9:23:CD:59	127.189.253.108/24	relay4.example.com
eth0	00:07:E9:23:CD:59	127.189.253.111/24	-
eth0	00:07:E9:23:CD:59	127.189.253.102/24	relay2.example.com
eth0	00:07:E9:23:CD:59	127.189.253.100/24	relay6.example.com
eth1	00:07:E9:23:CD:58	11.77.6.1/24	-
eth1	00:07:E9:23:CD:58	11.77.6.51/24	relay2.mail.example.com

Limitations and future work

Limitations

- So far, very limited number of patterns.
- So far, practical demonstration of utility in only one scenario.
- *Rich CI relations* and *multi-value attributes* address specific weaknesses of common SQL-based CMDB data models. May become obsolete with better out-of-the-box CMDB data models.

Future work

Patterns are “best practice”: Spread the idea and encourage participation

- Promote adoption of *rich CI relations* and *multi-value attributes with* CMDB tool developers.
- Promote use of patterns by other CMDB practitioners (e.g. by integrating them in a FitSM Guide).
- Gather feedback and new ideas.
- Develop further patterns and refine existing ones.

Conclusion

- CMDBs are critical for ITSM, but difficult to implement successfully
- A one-size-fits-all silver-bullet solution for CMDB design is very far away

- Patterns can serve as a toolset to aid in CMDB design
  - Common language for talking about CMDB data and information modeling
  - Building up a stock of reusable designs
- The three exemplary patterns presented…
  - aim to reduce complexity of the CMDB, while maintaining or enhancing utility;
  - do this either by reducing the number of CIs and relationships or by (to achieve a maintainable, easily visualized and understandable model)
  - Are general design patterns, but can serve as a basis for more specific patterns and CI templates (for common components, bits of infrastructure design etc.)

References

[1] *ISO/IEC 20000-1:2011 – Service Management System Requirements*, ISO/IEC, April 2011

[2] *ITIL Service Transition – 2011 Edition*, Cabinet Office, 2011

[3] Colville, R.J.: *Seven Steps to Select Configuration Item Data and Develop a CMDB Project That Delivers Business Value*, Gartner, January 2013

[4] Colville, R.J.: *CMDB or Configuration Database – Know the Difference*, Gartner, March 2006

[5] Adams, P. and Govekar, M.: *Hype Cycle for IT Operations Management 2013*, Gartner, July 2013

[6] England, R.: *ITIL's dead elephant: CMDB can't be done*, IT Skeptic (http://www.itskeptic.org/cmdb), June 2006

[7] Fowler, M.: *Analysis Patterns: Reusable Object Models*, Addison-Wesley, November 1996

[8] FitSM-5:2014: Guidance on the application and implementation of lightweight service management in federated IT infrastructures, FedSM project, http://www.fedsm.eu/fitsm, February 2014