Outline

• History
• ServiceNow CMDB – an overview
• Our approach to establishing the CMDB
• Changes since go-live
• Lessons learned
History

We went live with ServiceNow in October 2011 with:

- Tasks
  - Incident
  - Problem
  - Change
  - Request
- CMDB (c. 350K CI’s)
  - Hardware CI’s
  - Some software, applications, databases
  - Service offerings
- Other
  - People (users)
  - Groups
  - Locations
History

Since then...

- Expanded the CMDB from 350K to 700K
- Added Project & Release
- Onboarded many additional services
- Added functionality that builds on the CMDB
- ISO 20K Certification
  - Certified in 2012
  - Re-certified in 2015
## Some Current CMDB Stats

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware Assets</strong></td>
<td>70,000</td>
</tr>
<tr>
<td>Computers</td>
<td>30,500</td>
</tr>
<tr>
<td>Network Gear</td>
<td>3,200</td>
</tr>
<tr>
<td>Printers</td>
<td>550</td>
</tr>
<tr>
<td>Storage Devices</td>
<td>1,500</td>
</tr>
<tr>
<td>HEP Equipment</td>
<td>29,000</td>
</tr>
<tr>
<td>Other/Unknown</td>
<td>4,700</td>
</tr>
<tr>
<td><strong>Database Instances</strong></td>
<td>650</td>
</tr>
<tr>
<td><strong>Application Instances</strong></td>
<td>1,150</td>
</tr>
<tr>
<td><strong>Software packages</strong></td>
<td>130,000</td>
</tr>
<tr>
<td><strong>Service Areas</strong></td>
<td>50</td>
</tr>
<tr>
<td><strong>Service Offerings</strong></td>
<td>313</td>
</tr>
</tbody>
</table>

**Total # of CI's** 700,000  
**Total # of active CI's** 500,000  
**Total # of CI relationships** 25,000
ServiceNow CMDB
CMDB CI Relationships

- Application Suite
- Service Area
- Application Instance
- Database Instance
- Server
- Service Offering

Relationships:
- Depends on :: Used By
- Runs :: Runs On

Diagram shows the dependencies and relationships between different components in a CMDB environment.
Relationships between CI’s and ITSM processes

Service Offering

Application Instance

Database Instance

Server

Outage

Change

Request

Incident

Description

Configuration Item

Start

End

...
Service Metrics and KPI’s

- Service Commitments allow us to measure service quality
How did we go about establishing the CMDB?
Goals

- Have enough ‘stuff’ to support routine incidents and requests
- But: Let the Incident / Change/ Request process work even if the CI is missing or is inaccurate.

- Grow over time
  - More CI’s
  - More attributes
  - More classes

- Co-exist with existing business processes as much as possible.
  - Don’t change more than is necessary

- Strive for current and accurate info in the CMDB
  - Automate as much as possible
  - Make it easy for service providers to maintain the data
  - Give value to service providers (in return for the work they do to maintain the data)
Guiding Principles

• CMDB is there to support all aspects of service operation, not just change.
  • *Non-production items belong in the CMDB*

• Strive to preserve purity, simplicity and integrity of data within ServiceNow, but recognize that compromises will have to be made to co-exist with other systems or to meet our own unique business requirements.
  • *Customization is OK, but do it wisely*
  • *Follow ServiceNow’s naming conventions*

• Prototype early and often to validate assumptions and get ideas.
  • *Involve service providers and SME’s. The CMDB is for them.*
  • *It should ‘make sense’ to the providers and SME’s.*

• **Plan for how you will maintain the data**
  • *Integrations keep frequently changing info up-to-date.*
  • *Service Providers update CI’s and CI relationships as part of the Change process.*
Initial CMDB

− Hardware
  − Computers
  − Network Gear
  − Storage Devices
  − Printers
  − Scientific equipment
  − Other
− Service Offerings (from the provider point of view)
  − On-boarded departments -> ‘Service Areas’
  − Service Offerings within each Service Area
− Applications
  − Application Instances
  − Applications
  − Application Families
  − Database Instances
− CI Relationships

Where did the data come from?

Integrations

Interviews (Manual load)

Spreadsheets (Manual load)

Manual entry
### All Together Now...

#### Detected Info

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detected name</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux(Redhat)</td>
</tr>
<tr>
<td>OS Service Pack</td>
<td></td>
</tr>
<tr>
<td>OS Version</td>
<td>2.6.9-106.ELsmp</td>
</tr>
<tr>
<td>Detected serial number</td>
<td></td>
</tr>
<tr>
<td>Detected Model</td>
<td>PowerEdge 2850</td>
</tr>
<tr>
<td>Detected username</td>
<td></td>
</tr>
<tr>
<td>Detected description</td>
<td></td>
</tr>
<tr>
<td>CPU speed (MHz)</td>
<td>3,392</td>
</tr>
<tr>
<td>RAM (MB)</td>
<td>4,051</td>
</tr>
<tr>
<td>Disk space (GB)</td>
<td></td>
</tr>
<tr>
<td>CPU count</td>
<td>4</td>
</tr>
<tr>
<td>CD Speed</td>
<td></td>
</tr>
<tr>
<td>CPU manufacturer</td>
<td>Dell Computer Corporation</td>
</tr>
<tr>
<td>CPU type</td>
<td>Intel(R) Xeon(TM) CPU 3.40GHz</td>
</tr>
<tr>
<td>Detected by</td>
<td>OCS</td>
</tr>
</tbody>
</table>

#### Contract, Warranty and PO Info

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Contract Type</td>
<td>SVC PROV UNDER CONTRACT</td>
</tr>
<tr>
<td>Support Company</td>
<td>DELL 7X24</td>
</tr>
<tr>
<td>Contract Start</td>
<td>2012-05-14</td>
</tr>
<tr>
<td>Support Contract Description</td>
<td>HOTLIST</td>
</tr>
<tr>
<td>PO number</td>
<td></td>
</tr>
<tr>
<td>PO Date</td>
<td></td>
</tr>
<tr>
<td>PO cost</td>
<td></td>
</tr>
</tbody>
</table>

#### Support and Ownership Info

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custodian</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>FL/CCD/ESO/USS</td>
</tr>
<tr>
<td>User group</td>
<td></td>
</tr>
<tr>
<td>Asset Owner</td>
<td></td>
</tr>
<tr>
<td>Asset Owning group</td>
<td>FL/CD</td>
</tr>
<tr>
<td>Primary Admin</td>
<td>Linux Server Support</td>
</tr>
<tr>
<td>Support group</td>
<td>8bi17by6</td>
</tr>
<tr>
<td>Critical support schedule</td>
<td></td>
</tr>
</tbody>
</table>

---

Registered

Detected

Observed

Supported

Purchased

Deploying a CMDB
After Go-Live...
So: How did the adoption of the CMDB go?

• Users almost never see the CMDB.
  • We rarely ask for CI’s explicitly on requests and incidents
  • But we try to automatically associate the CI with each task behind the scenes.

• Service Desk routes fewer incidents and requests manually.
  • If the task ‘knows’ what CI it’s for, we already know who the support group is.

• Many service providers look at CMDB data more often then they ever did before.
  • They like seeing data from different sources in one place.
  • They were able to give up some spreadsheets.
  • They can find out who depends on their ‘stuff’.
  • They now expect good data. Spot and report inconsistencies and bad data.
  • They ask for more data - new classes, new attributes.

• We get a lot of demand from service providers for functionality that builds on the CMDB.
  • People are thinking in CMDB-terms, and they want more.
After go-live...

- We have been able to add many new functions and automations that act on CMDB data
  - Auto-expire BYOD devices that haven’t been seen on the network
  - Reports to managers about ‘stuff’ left behind when an employee leaves
  - Notifications to providers when certain things change on their CI’s
  - Monitoring systems auto-create incidents and associate them with CI’s
  - ‘Grants’ – permissions associated with CI’s that expire after a certain amount of time
  - Application events that can auto-create incidents
  - Website registration system with content reviews and Grants
  - EA Enablers – a way to make sense of the CMDB from the EA perspective
  - ...
Lessons Learned
Lessons Learned – What Went Well

Planning for CI Classes and Attributes

– Someone is in charge of the CMDB to make sure data integrity and design principles are preserved.
  
  • No new classes or attributes go into the CMDB without design review
  • This is a different function than Configuration Management

– Practical considerations:
  
  • We try to populate only ‘leaf’ classes. It makes reporting a lot simpler.
  • Create placeholder classes for unknown/other.
  • Add classes and attributes as needed. Make it look as close to real-life for service providers as possible.
  • But: think hard about where to add them.
  • Follow naming conventions
What Went Well

Planning for Integrations

– Methods:
  • ODBC via MID server (Scheduled Import)
  • Flat files from file shares (Scheduled File Import)
  • Manual spreadsheet uploads (CSV & XLS)

– Practical Considerations:
  • Identify Sources Of Truth / Authoritative Sources
  • Agree on one Authoritative Source
    – Per table/class (only one Authoritative Source creates CI’s per table)
    – Per attribute/column within table/class (only one Authoritative Source updates an attribute within a given table)
    – Avoid ‘updater wars’
  • Put effort into identifying the keys by which you will correlate CI’s
    – We found that MAC addresses work very well for hardware
Lessons Learned – What Went Well

Accommodating Service Providers

– Build trust
  • If the data is incorrect, they will not trust the CMDB
  • If data is missing, they will not use the CMDB

– Practical considerations:
  • Make it easy
    – As little manual updating as possible
    – Change is a good way to manage updates
  • Give them something they didn’t have before
    – Data from disparate sources in one place.
    – What will be affected by your change/incident?
    – What changed upstream right before the incident happened?
Lessons Learned – Challenges

• Our customizations sometimes went in a different direction than ServiceNow’s upgrades.
  • The Asset module came after we went live. We would have designed some integrations differently if we started out when Asset was in place.
  • We really want to use ServiceNow’s Asset module, but it will be a lot of work.

• Be cautious of first version of new ServiceNow modules.
  • We implemented Release v1.0 and customized it enough to work for us.
  • Then ServiceNow greatly enhanced their Release.
  • Then they built Demand on top of Release.
  • We really want Demand but it will be challenging to change how we use Release.

• Prototype a lot. If you made a design mistake in ServiceNow:
  • Fixing the design is often the easy part
  • Making people change what they are used to doing is hard
Questions?

Krysia Jacobs
kjacobs@fnal.gov