



Nine Steps to Implement a Successful CMDB Project

info@evergreensys.com
571.262.0977

Table of Contents

1. Education and Awareness	3
Definition.....	3
What's the Value?.....	4
What It's Not	5
How is a CMDB Unique?.....	5
CMDB and Asset Management	6
The Project	6
Key Success Factors	7
2. Prioritize CMDB Use Cases	7
3. CI Needs Assessment	14
CI Types.....	14
4. CI Design Decisions	16
The Jargon	16
Integration Approaches.....	17
Sourcing Decision	17
CI Key Matching	18
CI Attribute Level Mapping	19
Pick Lists	20
5. Discovery	20
Grant Security Credentials	22
Probe Installation	25
Services Activation.....	26
Initial Discovery and Baselining.....	26
Custom Discoveries	26
Scheduling	27
Probe Monitoring	27
6. CI Ownership and Duties	27
Formal CI Assignment to Domain Owners	27
Recurring Duties.....	28
Continuous Improvement	28
7. Relationship Mapping	28
Understand Configuration Relationships.....	28
Virtual Relationships	29
Define Mapping Scope	29
Enrichment	30
Correlation Severity	30
8. View Creation	32
View Examples	33
View Training.....	33
View Plan.....	33
Snapshots.....	34
Events	34
9. CMDB Administration Processes	35
Physical Maintenance	35
CI Type Change Control	36
Discovery Monitoring	36
User Access	36
Enterprise Views.....	36

Global Changes	37
Enhancement to Satisfy View Requirements.....	37
Appendix 1 - Roles and Responsibilities	38
Appendix 2 – Skill Set.....	40
CMDB Administrator	40
CMDB Discovery Manager	41
Mapping Manager.....	42
Appendix 3 – Current CMDB Vendors	44
Product Weaknesses to Avoid	44

Nine Steps to Implement a Successful CMDB Project

CMDB is a popular topic these days. Everyone needs it. Few can explain it. Fewer can justify it. ITIL highlighted the need for Configuration Management and the software vendors responded with products. Analysts added their endorsement using terms like 'federation' and 'reconciliation' to explain the gaps while also making a stab at explaining the benefits. Yet actual CMDB adoption has not kept pace with the market's interest in it.

Intuitively people accept that Operations needs tools to help make better decisions, and few would deny that IT is a technology-immature organization. The current discourse is reminiscent of the early data warehouse days where products and technology drove the discussion using terms like OLAP, ROLAP and data marts with a liberal dose of "build it and they will come" reassurance. It seems that most new software product waves only focus on features and ignore implementation guidance.

This whitepaper lays out a prescriptive nine-step plan describing how to implement a CMDB, including best practices. This whitepaper may be used to educate staff and design an effective CMDB project plan.

1. Education and Awareness

A CMDB brings a new and unique capability. New concepts have to be digested by the organization to change how IT thinks and makes decisions. Tools can be quickly installed but if the hard work of explanation and alignment are not completed then the tools it will remain toys in the hands of few.

A CMDB does not address the usual IT concerns of functionality, availability and performance but instead addresses how components are connected and constructed, as well as their dependencies upon other components. Like all operational tools, a CMDB is helping predict and prevent service disruptions by shortening problem resolution time and improving quality of service. However, a CMDB does add the new dimension of structure.

This section discusses concepts that need to be communicated and tested within the organization before implementing a CMDB.

Definition

Current definitions of a CMDB can be vague, for example:

- ❖ A CMDB implementation helps organizations adopt best practices.
- ❖ ITIL compliance requires a CMDB.

- ❖ A CMDB is a single source of record.
- ❖ CMDB is configuration management.

Consider this definition of a CMDB:

1. It is a decision support application that allows one to visualize what's there and how one may interact with what's there.
2. It tracks and reports on configuration events.
3. It formalizes components by assigning an identity, properties, and structure.
4. It can account for many types of components also known as Configuration Items (CIs) and their relationships. CI Types can be anything you believe supports making decisions ranging from networks, appliances, servers, software, rules, files, people, location, documents, etc.

What's the Value?

What can be done with a CMDB? Here's a sample of scenarios a CMDB can tackle:

- ❖ A problem has occurred. What has recently changed? Was it approved or not?
- ❖ A change is being contemplated. What are the end points, and who could be affected and in what manner? Also the reverse situation can be addressed when a problem CI needs to be researched. A CMDB can indicate what a CI is connected to and its interdependencies with other CIs.
- ❖ A dashboard is required to summarize what's currently happening by CI or by layer. A capability to drill down to the offending layers and child CIs is required, one that would call out details such as:
 - Calls/incidents and tier 1-2 and 2-3 escalations
 - Critical operational events
 - Utilization, capacity, SLA violations
- ❖ A summary is required that details what's happening to a CI over time and answers questions such as:
 - What is the total cost of ownership? Can support/change costs be used to negotiate with the vendor? Does it justify replacement?
 - Is MTTR getting shorter?
 - What is the change history?
- ❖ What is really installed now?
 - Return to a known state (test, DR, expand a farm)
 - Demonstrate compliance to internal or customer policies
 - Track if a configuration complies to its installation pre-requisites, either as specified by the vendor or the developers
 - Verify promotion or rollback or replacement

- How does a CI differ with a declared standard or between two similar CIs?
- Was a service catalog request fulfilled?
- ❖ If the CI is known how can related information be located?
 - Manuals and specifications
 - Asset information, e.g., ownership trail, maintenance contracts, lease terms
 - Physical location and moves, or for software, deployment points, consumed licenses, etc.

As a sidebar, if an SOA initiative is being pursued, a CMDB will by definition drive towards formalizing component architectures and implementing service registries using configuration governance. Although it's under the SOA rubric, it's the same configuration objective in managing structural integrity.

An analogy might help. A CMDB is the IT operations equivalent of a customer service crm system that provides a single 360 degree view of the same customer. Heterogeneous platforms and disparate data prevented customer-centric interactions and analysis. However, if all the interactions, such as fulfillment, support calls and up-selling, could be viewed as one logical customer then competitive advantage would result. A CMDB applies the same idea with CIs instead of customers.

What It's Not

By way of contrast, a CMDB is not/does not:

- ❖ A technically complex or special database
- ❖ A process-oriented application, like change, release or provisioning
- ❖ A monitoring application, although a CMDB may include selected system management events
- ❖ A source version control or library
- ❖ Have to be part of an ITIL program
- ❖ A Service Catalog, although a catalog can and should use the same CI structures
- ❖ A massive, all encompassing physical data store, although pointers can link a CI's attributes remote source systems
- ❖ A turnkey product

How is a CMDB Unique?

An organization has many database applications. A CMDB differs from a typical relational business database as follows:

- ❖ Records are populated from a discovery and mapping engine with minimal manual entry

- ❖ Data extraction and visualization typically does not use routine SQL but Query By Example or models.
- ❖ Table structure is oriented around abstract objects and internal event management and not a business process.
- ❖ Relationships are themselves objects and specific to CI affinities and not static relational table relationships. Relationships are more important than the data in the tables and are novel and specific to configuration management.
- ❖ End users interact with a CMDB in two distinct modes – requesting views and responding to configuration events.
- ❖ A CMDB has a proprietary but generalized table design. The CMDB is a simple database in terms of the end user data; however, the tables and procedures make it more complex as a result of managing events. End users should not directly manipulate a schema using SQL code or expect to see normalized tables. Fortunately the tables are extendable and accessible via APIs and as such, every effort should be made to use vendors' APIs, maps, and visualization. An alternate method to APIs is to use the discovery engine to do a data pull.

CMDB and Asset Management

Sometimes a CMDB is confused with Asset Management. Asset Management is not a CMDB. A CMDB exists to improve operational decisions. An Asset Management system exists to enforce financial and custodial policies as well as accounting for the CI throughout its lifecycle. A CMDB is mainly interested in the CIs in production whereas an Asset Management system handles other processes such as planning and procurement, physical and budget ownership, location, audit trail, location tracking, disposal, lease and maintenance terms. A CMDB might interface with an Asset Management to obtain operational attributes such as expiration dates, asset ID, age, maintenance contract info, location.

The Project

A CMDB is a project. This tool cannot be successfully implemented without some structure, planning and a lot of education. Although the initial setup and discovery can be completed in two to three months, the project itself will take a year. However, unlike most IT projects, a rapid concentrated effort is not desirable. It takes time to learn and apply what a CMDB can do to actual problems. The nine steps in this paper can be followed as actual phases in a CMDB project plan. Appendices 1 and 2 provide a detailed description of the project team and requisite skills.

The factors in the following table may provide a sense of proportion about the scope of a CMDB project.

Scales/Tiers	Small	Medium	Large	Enterprise
Hosts	1 - 999	1K - 10K	10K - 50K	50K+
Discovery Probes	1	2	3 - 5	6+
CIs in the CMDB	0 - 100K	100K-3M	3M-10M	10M+
Active View Nodes	0 - 1K	1,001-2K	2,001-3K	3K+
Mapped Applications	0 - 10	11 - 49	50 - 250	250+

Key Success Factors

While developing the project charter, consider adding the following success criteria:

1. IT personnel begin to think in terms of structure when solving problems.
2. All areas that use discovery technology or use CIs as their core data use the CMDB or the same conventions, e.g., application, network and enterprise monitoring.
3. CIs are owned by the person responsible for their performance in production. Ownership means they are responsible for configuration integrity.
4. The CMDB is used daily by almost every function in IT and becomes a shared trusted information source which can reduce search time and can highlight incremental changes.
5. As every problem is solved, key learnings should be captured. In addition to updating test routines, running books, monitors/alerts and documents, the CMDB should be updated with new CI types, attributes, relationships, views and events.
6. A CMDB Administrator is responsible for a limited set of ongoing support processes (see section 9).
7. A standard set of reusable views or maps are maintained and made available for new projects.
8. The CMDB is used for ad hoc analysis instead of manual research.
9. Discovery can automatically source 99% or more of the CIs without manual intervention.
10. Relationship mapping supports all of the desired use cases and potentially spans from routers to end users over time.

2. Prioritize CMDB Use Cases

A CMDB supports a variety of technical analyses. A target set of Use Cases or report requirements should be selected during the typical requirements phase. However, defining requirements in the traditional sense is problematic for a CMDB. Some of the barriers are:

- ❖ Users will not be familiar with CMDB functionality.
- ❖ The data, i.e. CIs, has never been formalized so some of the terms are not part of the language, for example Customer ID.

- ❖ A CMDB is often used in an ad hoc manner similar to mining data warehouse data and it will take time to begin looking at the environment and solving problems using configuration information.

While the expectation should be set that the first cut of requirements will not be rigorously defined, some analysis and prioritization should be attempted. These scenarios or User Cases are simplistic and not true UML process modeling with actors, sequence, etc. The most common Use Cases are graphical and two dimensional views. Views can be subdivided into:

- ❖ General views designed around user's interest area
- ❖ Event driven (or highlighted) views
- ❖ Ad hoc views which are configured on the fly in response to a question or problem solving session

An implementation may also encounter requirements to integrate certain configuration events into other processes such as discovery of a new CI. Another approach could be to embed CMDB queries into external forms and workflows.

The Use Cases from this step should (1) validate that the required CIs are being addressed in the CMDB and (2) guide how the first set of standard views and maps will be configured.

Use Cases can be configured around the following topics:

- ❖ Application Mapping
- ❖ Fault Management
- ❖ Monitoring
- ❖ Root Cause Analysis
- ❖ Impact Analysis
- ❖ Data Center Relocation/Consolidation

More specific examples of Use Case functions include:

- ❖ Review of recently changed CIs
- ❖ Identification of unapproved changes
- ❖ Review of open High Priority Incidents, Calls, Problem by CI
- ❖ View key operational states, events and alerts for CIs
- ❖ Collect CI Total Cost of Ownership
- ❖ Verify task completion from either a service catalog fulfillment or change actions
- ❖ Locate key documents by CI
- ❖ Production Run Book view

- ❖ Compliance views, internal or customer-specified
- ❖ Access change history trail in support of diagnosis
- ❖ Evaluate impacts for proposed changes
- ❖ Verify rollback to the last known consistent state
- ❖ Compare DR or consolidation to current production environment
- ❖ Security analysis of penetration points
- ❖ Verify network redundancy, clustering
- ❖ Discover servers, end points, applications, network devices for Asset Management purposes
- ❖ Scan for unapproved software
- ❖ Verify configuration settings and existence of key parameter files
- ❖ Views for vendor software support calls

Each organization needs to consider what views and configuration rules make sense for their business and environment complexity. Some open-ended facilitation questions that can be used are:

- ❖ If x was known how could y have been prevented?
- ❖ What catches the organization by surprise?
- ❖ What can others do that can adversely affect your area (and the reverse)?
- ❖ How can visualization aid in explaining complexity?

At this point it may help to review a few sample maps and table reports:

- ❖ Basic and more complex ERP application map view
- ❖ Server Comparison Report
- ❖ Compliance Report
- ❖ Change Auditing Report

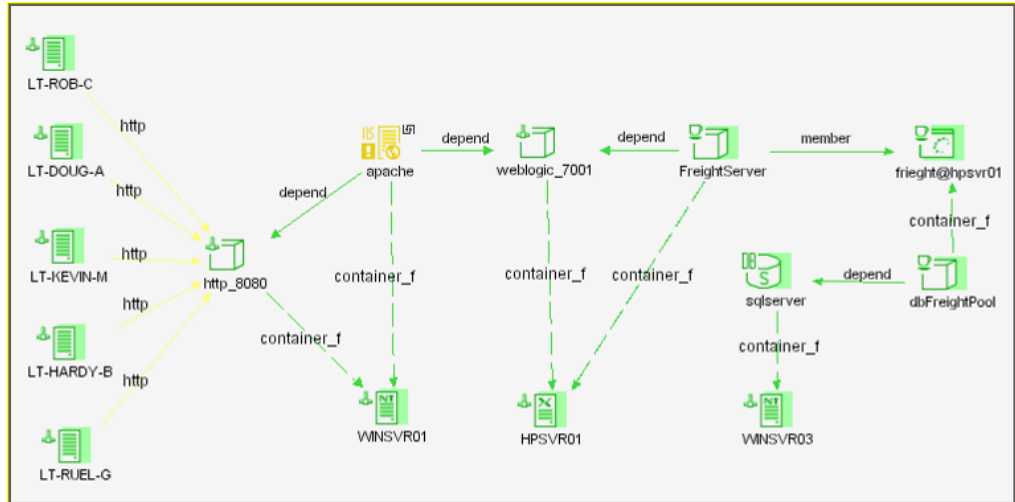


Figure 1 - Application View

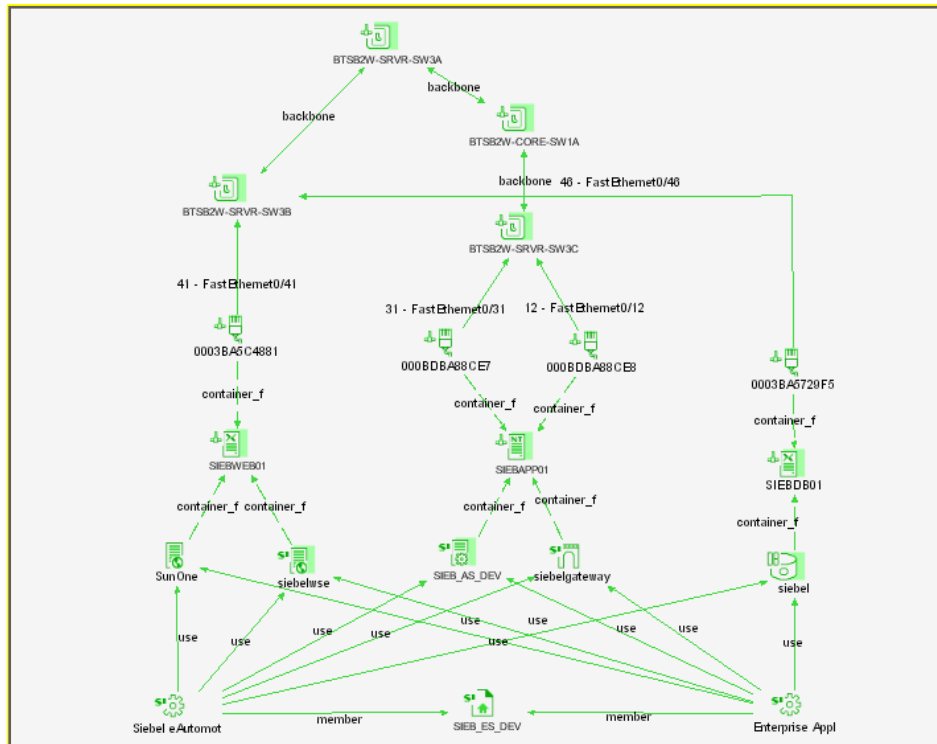


Figure 2 - More Complex Siebel Application View

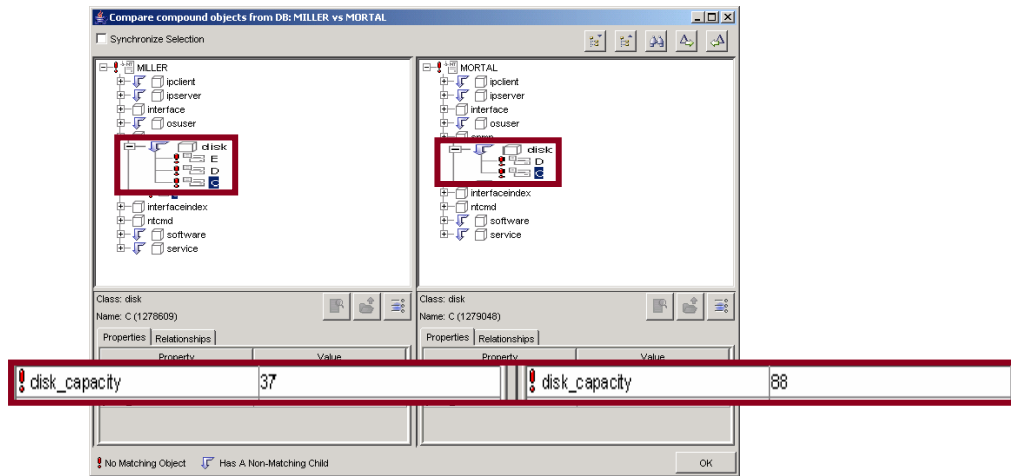


Figure 3 - Server Comparison Report

Gold Master Object - PC-PATRICK-D				
Report Time -				
LT-JIM-O				
			Missing	Additional
Component	Parent Object	Count/Gold	Missing	Additional
cpu	LT-JIM-O	4/2	cpu(cpu_speed:1393,cpu_vendor:GenuineIntel) cpu(cpu_speed:1393,cpu_vendor:GenuineIntel)	cpu(cpu_speed:1794,cpu_vendor:GenuineIntel) cpu(cpu_speed:1794,cpu_vendor:GenuineIntel) cpu(cpu_speed:1794,cpu_vendor:GenuineIntel) cpu(cpu_speed:1794,cpu_vendor:GenuineIntel)
disk	LT-JIM-O	7/5	Virtual Memory (disk_used:923.00903,disk_capacity:22,disk_size:4132.41) D:\ Label: Apps Serial Number 5c6110f0 (disk_used:3554.3164,disk_capacity:2,disk_size:14019.86) C:\ Label: Serial Number 7c7b179d (disk_used:6231.222,disk_capacity:96,disk_size:6448.49) D: (disk_used:3554.3123,disk_capacity:2,disk_size:14019.86) C: (disk_used:6231.222,disk_capacity:96,disk_size:6448.49)	Virtual Memory (disk_used:456.26163,disk_capacity:11,disk_size:4131.2583) Z:\(disk_used:0.0,disk_capacity:0,disk_size:0.0) D:\ Label: Serial Number a0819483 (disk_used:1706.0173,disk_capacity:1,disk_size:138998.97) C:\ Label: Serial Number 84a1d74a (disk_used:3741.551,disk_capacity:58,disk_size:6448.587) A:\(disk_used:0.0,disk_capacity:0,disk_size:0.0) D: (disk_used:1706.0167,disk_capacity:1,disk_size:138998.97) C: (disk_used:3741.551,disk_capacity:58,disk_size:6448.587)
interface	LT-JIM-O	4/2		000DBA88CD9(interface_description:null) 000DBA88CD8(interface_description:null)
ipclient	LT-JIM-O	1/3	nbession[10.49.61.101](Name:null,ipport_number:0) orada[10.49.61.139](Name:null,ipport_number:0)	
ipserver	LT-JIM-O	7/7	GENESYS_OutboundDBServer_4001 (Name:GENESYS_OutboundDBServer,ipport_number:4001)	ftp_21(Name:ftp,ipport_number:21)
memory	LT-JIM-O	1/1	6131952MB(memory_size:6131952)	6131108MB(memory_size:6131108)

Figure 4 - Compliance Report








Category	Class	Label	Msg	Ack	Discovery Time	Corr	Origin
 change	nt	WINSVR03	Object WINSVR03 attribute nt_servicepack was changed to Service Pack 3		2005.01.31_15:31:55		change
 change	nt	LT-MIKE-C	Object LT-MIKE-C attribute nt_servicepack was changed to Service Pack 2		2005.01.31_15:37:48		change
 change	nt	PC-CHRIS-A	Object PC-CHRIS-A attribute nt_servicepack was changed to Service Pack 2		2005.01.31_15:38:18		change
 change	nt	LT-TIM-R	Object LT-TIM-R attribute host_snmpsysname was changed to LT-TIM-A		2005.01.31_15:38:37		change
 change	unix	MERQSVR0072	Object MERQSVR0074 attribute host_snmpsysname was changed		2005.01.31_15:39:50		change
 change	unix	MERQSVR0079	Object MERQSVR0076 attribute host_snmpsysname was changed to MERQSVR0079		2005.01.31_15:40:05		change
 change	switch	BTSB1-LAB-SW3G	host_snmpsysname was changed to BTSB1-LAB-SW3G		2005.01.31_15:40:48		change

Figure 5 – Change Auditing Report

3. CI Needs Assessment

Based on the Use Cases certain data decisions must be made. As with User Cases these are not strict and exhaustive requirements to design a business database. The CMDB's embedded discovery tool provides a generic set of CI Types. This can be accomplished by the discovery engine, through collecting CIs and then evaluating its adequacy. However, the tool's acceptance can be advanced if operations are considered and information is included specific to the environment. Analyzing and defining CI Requirements involves four steps:

- ❖ CI Type and Attribute Needs Analysis
- ❖ Source System Identification
- ❖ Enrichment
- ❖ History

CI attributes are merely related fields or properties. For example, if a disk drive is a storage CI Type, then the maximum formatted capacity would be one attribute.

CI Types

A CI Type is a class of objects that share similar functionality and purpose. Types are used for convenience and are not really implemented as a true polymorphic class in an object-oriented sense. A host is a common CI Type. A host provides resources for software to execute. An IP Phone, Linux server and laptop would be different types of hosts. A CI is an instance or actual object. A CI does not have to be limited to physical tangible objects. Routing rules, documents, content, SLAs, etc., can be valid CIs.

A CI Type analysis can be evaluated in two dimensions. The first dimension is the 7 Layer ISO Protocol Stack that can form the rows or CI Types. The columns identify CI Type attributes that result from considering processes and related source systems.

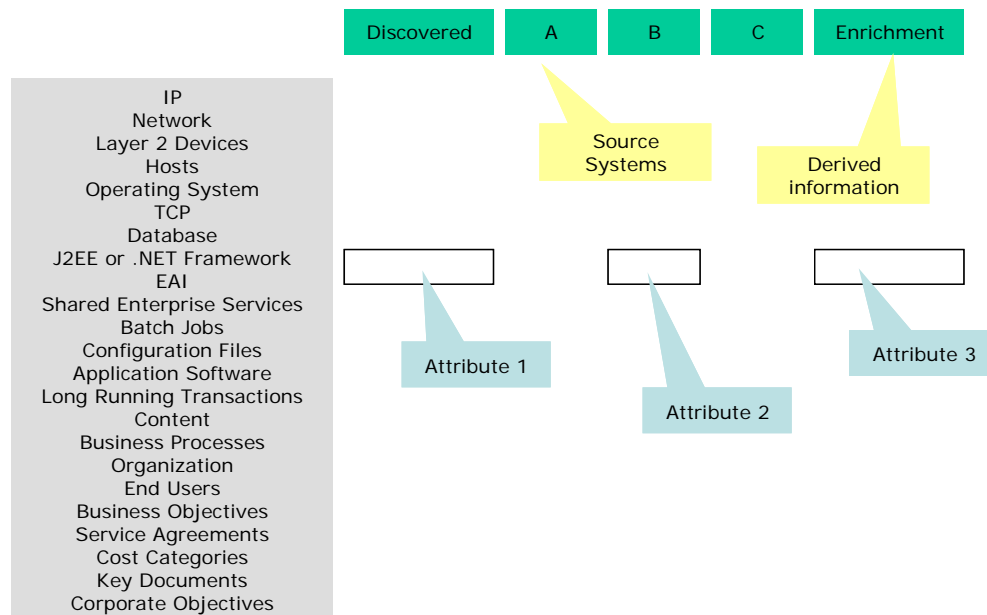
In addition to the Use Case views candidate CI Types can also be identified from the following sources:

- ❖ Existing monitoring tools
- ❖ Run book
- ❖ Installation guides
- ❖ Technical policies
- ❖ Prior test defects
- ❖ Change history
- ❖ Source and version control
- ❖ Provisioning tools
- ❖ Problem/root cause analyses
- ❖ End-user input
- ❖ Existing diagrams

A CMDB will often share CI attributes with the following tools:

- ❖ Asset Management
- ❖ Service Desk and ITIL Processes
- ❖ Application Monitoring
- ❖ Security/Network Management
- ❖ Schedulers

The following diagram illustrates this at a high level. The CI Types depicted would need to be further subdivided to be practical. When matrixed to other systems like Asset Management it would show that certain attributes should be shared like Asset Tag ID, purchase date, license expiration dates, annual lease cost, vendor maintenance and current physical location.



4. CI Design Decisions

Some data integration will be necessary. During the CI Needs Analysis candidate Source Systems should be identified. This step defines what and how data will be shared.

The Jargon

The analysts and software vendors have introduced vague and unfamiliar language into the CMDB topic. These terms are not well defined and today the technical solution is incomplete. This complicates explaining how a CMDB functions and raises concerns that some magic is required, which is not the case. The problematic terms are:

Federation	...bringing in multiple data sources directly by linking to sources
Reconciliation	...avoiding duplicates and enabling matching of configuration items from different sources
Synchronization	... identifying changes to the CMDB as they occur thereby ensuring the same version of the truth across integrated systems

In terms of CMDB coexistence or integration, the only solution currently available is an API along with an SDK with query and update services functions. One reason is the lack of any Operations data standards, although some have been attempted. Another is that software vendors do not have the knowledge or positioning to design an enterprise solution in an immature and fragmented data environment.

So the reality is that approaches like Federation really mean that the "customer has to figure out the gaps". On the other hand CMDB integration is not that complicated. The data is sparse and a lot of latitude is granted in selecting when to add CIs and their level of granularity, similar to a data warehouse. In no way is it a transactional or real-time application. A review the terms and more specific analysis should illustrate the point.

Federation

This is not a useful term or a realistic expectation. The assumption is that you can put pointers into the CMDB and link to content in external systems. The problem is that the CMDB does not directly render external content and source systems may not have a suitable API. Of course you can place hyperlinks and launch sessions to display information if the source system supports that but that's not true integration.

Reconciliation

Duplicates are inevitable because different discovery patterns will find the same CIs and attributes. Part of the answer is in purchasing a CMDB that has a prioritization scheme. The CMDB Administrator can also minimize duplication by assigning one system of record for each CI Type and restricting/filtering the discovery engine to this system. Other than that there is no special reconciliation function to implement.

Synchronization

Synchronization is managing asynchronous data concurrently. CMDBs are not transactional systems so true real-time synchronization or distributed commits are not practical. However, this is also not a major problem because a CMDB is a decision support tool. The potential for serious data quality issues can be avoided by establishing that the CMDB will control the base CI records and primary keys and external source systems will only be used to add attributes or subordinate CIs.

In summary, the jargon can be ignored. Just remember that:

- ❖ a CMDB is more like a data warehouse than a transactional system
- ❖ the extent of data integration is determined by you the customer
- ❖ the integration needed is no more complex than any database-level integration effort

Integration Approaches

Despite the lack of technical specifics there is still a need for a mechanism to share data. Three options are identified. Because there is no universal solution, one can feel free to design an individualized approach as well. For the first implementation phase the CMDB is usually integrated with the Asset Management or Service Desk (for calls, incidents and problems).

Discovery Initiated

A recommended approach is to “find” adds, changes and deletes using the discovery engine as a query mechanism. This enables fine control over the update frequency via the discovery schedule. It also makes it easier to control which systems provide which CIs. Most discovery engines will already handle SQL to discover database objects so this is merely another variation. Discoveries will have to be customized to the source systems but they should be straightforward SELECT or UPDATE statements. However it should be kept in mind that this is a pull mechanism. Events and any time-critical changes still need to be pushed to the CMDB using its update API.

Integration Tools

If your organization has access to either EAI (i.e., translation and routing) or data warehouse ETL (Extract Translation and Load) tools, then these can be leveraged for replicating external data into/from the CMDB. However it is still recommended to use the CMDB APIs since directly manipulating the CMDB tables is not a good option.

Brute Force

The standby option is to construct batch update programs or web services to perform the integration. This requires some coding but may be the most comfortable approach, at least for the initial implementation.

Sourcing Decision

After determining the integration complexity and budget, the effort returns to developing a design to populate the required CIs. The first step determines if the CI and attributes will be obtained via the discovery engine directly, manually entered or from an external database. It is advisable to highlight where the same data exists

in other systems. The first two columns should match the worksheet developed in the CI Needs Analysis section. Only a sample of data is shown here. It may be necessary to insert a column to subdivide CI Types into categories that are not enterprise-wide or exist in specific domains and places.

CI Types (Subset)	CI Attributes	Discovered	Manual	Source System		
				Asset Mgmt	System B	System n
IP		●				
Network		●				
Layer 2 Devices	Asset Tag ID			●		
	Purchase Date			●		
Hosts	Asset Tag ID			●		
	Purchase Date			●		
Operating System	License expiration dates			●		
Application Software	License expiration dates			●		
Service Agreements	Annual lease cost			●		
Key Documents	Vendor Maintenance			●		

CI Key Matching

The next step defines the CI at a record level. The Source System of Record must own, i.e., assign the primary key, add and delete the record. The CMDB and Source Systems must maintain mutually recognized data to support a lookup. Push/Pull defines who initiates the transfer. Triggering event specifies a real-time event, scheduled task, etc. Version Control can be a sequence field, timestamp, update number or always overwrite.

Some CIs need their events to be logged. Logging maintains a historical trail for a CI and should be used on a selective basis where it improves the resolution of problems. Logging is a built-in CMDB feature and is not the same as transaction logging. This is usually driven by either policy or technical risk. Risk is composed of factors like the number of dependencies, problem history, worse case impact, etc.

In a CMDB the relationships are objects that associate CIs and not relational table relationships. The specific CMDB will dictate how much helping information should be provided to maintain CI mapping.

Be aware that the discovery engine will automatically calculate a primary key. If it's a device then a combination of the MAC address and DNS names is typically used.

	Primary Key	Alternate Key(s)	Push/Pull	Trigger	Version Control	History
CIs						
Relationships						

CI Attribute Level Mapping

The next step in the process is to describe the field mapping. 'C' means the CI is 'Created' in the CMDB from the source system, and 'E' means the CI 'Extends' the attributes in a CI already residing in the CMDB.

'CMDB Tag' is the reference to a form display or field name in the CMDB. Some attributes are used for CI relationships and not viewed by the CMDB user. These tend to be used for CIs like documents, rules, people and other situations where the relationship is derived and not directly observed or inferred by the discovery engine.

For instances where the CI attribute is a reference to another system, create some nomenclature to inform the user that this is a reference link and include additional attributes that assist the user in searching for and locating the CI. Naturally, hyperlinks can also be used if the source system is reachable via a browser.

Be sure to provide one name attribute that is compact and readable for large and dense graphical maps. For example, a fully qualified DNS name will be too large and may clutter the reference.

Most fields in the CMDB are VARCHAR, so physical formatting rules do not have to be strictly observed.

		CMDB			
		Action	CMDB Tag	Used for Relationship	Translation or Reformatting
Asset Management	Asset Tag ID	E	Asset #	No	None
	Purchase Date	E	Acquired	No	Strip time out
	License expiration dates				
	Annual lease cost				
	Vendor Maintenance				
Infrastructure Monitoring					
Application					

		CMDB			
		Action	CMDB Tag	Used for Relationship	Translation or Reformatting
Monitoring					
Directory					
Security					
Users					
Service Desk					
Change Process					
Provisioning					
Network Monitoring					

Pick Lists

In addition to record-level integration, ensure that the analysis also identifies external systems that need catalogs of CIs. Processes and tools like the Service Desk need to create incidents, problems, changes and releases using a standardized CI naming convention. Bulk loads are adequate for updated CI lists.

5. Discovery

Automated discovery is the second most critical function in a CMDB. It is absolutely essential that the right discovery engine is selected. A variety of tools perform limited discovery, like network monitoring, asset/inventory applications, desktop provisioning and perhaps SMS. A CMDB, on the other hand, must discover a much larger range that goes literally from wires to people, otherwise it will not be rich enough to support true analysis and will be just another server catalog.

The discovery engine must discover the environment automatically with absolutely minimum manual data maintenance, preferably using one discovery engine (i.e. some vendors require multiple discovery engines). Too many changes occur throughout the IT universe to manually track and update them. Implementing the discovery phase involves eight steps:

- ❖ Probe installation
- ❖ Granting security credentials
- ❖ Services activation
- ❖ Baselining
- ❖ Virus scanning
- ❖ Services cleanup
- ❖ Custom discoveries
- ❖ Scheduling
- ❖ Probe monitoring
- ❖ Enhancement to satisfy view requirements

Of the available discovery technologies, only agentless, credential-based discovery is adequate; therefore, the approach described in this section will address that method. In the rare instance that real-time information is needed, then agents can be selectively used to complement them. Perceived installation complexity or assurances about no maintenance are secondary to the ability to discover all the required information without modification.

Three main discovery techniques are available. Naturally, each has tradeoffs. 'Agentless' utilizes a software-based probe and uses credentials to collect CI data according to a specified schedule. Agents are software applications or services that are installed on the target host and granted privileges to monitor and sometimes perform updates. 'Passive' uses a network tap that scans traffic patterns and makes inferences based on rules. This is similar technology to real user monitoring.

Capability	Agent-less	Agent-based	Passive
Ability to reach detailed CIs	Yes	Yes if device has agent	No, limited to what can be sniffed and/or inferred from network traffic
Customizable discovery	Yes	No	No
Quickly adjust to new view requirements	Yes	No, capabilities are fixed in agent design	No, new rules or inference patterns must be developed
Can use for Disaster Recovery or other comparative analysis	Yes	Yes	No, can only detect what is actively running on a network
Real time events	No but can adjust discovery schedules	Yes	Yes if events are supported
Off line events	No	Yes	No
Network load	Minimal	Minimal	None
Access control	Credentials for services	Credentials	None
Audit what discovery probe is doing	Yes	Sometimes	No
Can trigger adverse events	Yes, virus scanning has to be adjusted; some network devices have SNMP defects that are triggered by repetitive queries	Yes but usually only during initial configuration	No
Skill level required to maintain	Moderate	Low	High
Cost	High	High	Low
Ease in Installation	Moderate		Simple

A common concern voiced about agentless discovery is that it consumes excessive network bandwidth. In reality the amount of data transmitted is very small, and standard services have small packet payloads. Also some engines have a backoff feature to avoid exacerbating slow devices or timeouts. The point is that CI data is sparse and equivalent to about 10 web pages unless you decide to pull every CI

from a Weblogic or Oracle instance. In addition, the discovery schedule is under management control so longer operations, such as sweeping for new IPs, does not have to be done frequently or can even be done on demand. The larger problem is that the target services may never really been configured and are merely installed with all defaults. If concern still exists, then the team should be encouraged to attach some monitoring and directly measure the bandwidth consumed by each discovery routine.

Grant Security Credentials

Two concerns are often raised with agentless or credential-based discovery. Usually they are reactions due to poor communication or bad assumptions. Some common objections are:

1. Security credentials are a pain to get.

Regrettably this is usually true, especially where security is loosely administrated or political issues are anticipated. The answer is to not keep the Security group in the dark during the planning phase, even if they don't understand the CMDB purpose. Without explaining the context up front, discovery by definition can easily be interpreted as antithetical to the whole notion of security. The rationale to explain is:

- Discovery requires access to systems to fulfill a valid purpose like any other user and the need is determined by the users of the CMDB and owners of the CIs.
- These credentials are usually based on defined services like JMX, SNMP or WMI and are probably already being granted today.
- Security should already have domain accounts with read-only administrative privileges that the discovery engine can join.
- If, say, a UNIX farm does not have administratively robust security management and, as a result, has to add IP addresses manually, then that should not be seen as a problem due the discovery engine.
- Credentials should be granted with the option to monitor, audit, turn off quickly and explicitly account for discovery access in firewall and virus rules.

2. Having all these credentials in one place in dangerous.

- First, it's also a valid argument that centralizing enables better control.
- The selected CMDB product must provide for encrypted database tables and transmission or reverse proxies because the credentials do, in fact, need to be stored. Credentials should only execute in memory and not persist.
- Keys used by the CMDB must be reissued and stored outside of the vendor's default installation routine.

- Only the necessary privileges need to be granted, e.g., SNMP credentials need to be able to read part of the OID tree but it does not need to execute management functions that SNMP allows.
- The CMDB itself needs to provide role-based security so access to administrative functions can be limited.
- Someone from security should enter the credentials into the CMDB/discovery engine and not the CMDB administrator.

It may be helpful to organize this information using the sample worksheet. CI Type comes from the Needs Analysis section. To this is added the protocol, level of access, etc., to help organize gathering credentials. This is only partially populated to show representative data. Each vendor has its own approach. Because there are many-to-many relationships between some CI Types and Access Protocols, this step should include selecting which protocol should be used to populate CI Types.

However, it is not necessary to create a one-to-one relationship. A CMDB should have the capability to resolve and prioritize data for one CI from multiple protocols. Each environment is different and some protocols, like Telnet, may be unavailable by policy. The following partial table shows commonly required credentials.

CI Type	Access Protocol	Needed Privileges	Grantor	Varies By
			(Name)	E.g. Physical Network, Mgmt Domain, Other
IP	ICMP	NA		
Router Concentrator Switch Port	SNMP	Get Read Only		
Host IP Interface Network Bridge Port Layer 2 links Backbone Links Route links Bridge links	SNMP	Get Read Only		
	Telnet	Regular (nonroot) OS user		
	SSH	Regular (nonroot) OS user		
	WMI	Admin User or enable WMI permission for a regular user with: Execute Methods, Provider Write, Enable Account for CIM2		
Server Inventory Disk Printq Program Service Software OS user CPU	SNMP NETBIOS Telnet SSH Monitoring Events	Same		

etc				
TCP Connections IPserver IPclient TCP links etc	SNMP NETBIOS Telnet SSH	Local Admin User		
Microsoft Domains	WIN API			
DB2	SQL	db2user		
Oracle dbaobjects dbarchivefile dbclient dbcontrolfile dbdatafile dbextent dbindex dbjob dblinkobj dbredofile dbsegment dbsnapshot dbtable dbtablespace dbuser owner program	SQL	DB user with "READ" permission from V\$ and DBA_ tables		
SQLServer sqldatabase sqlbackup sqlalert sqljob sqljobstep sqlperformance monitor sqlprocesses program Datasources: sysprocesses sysdatabases backupset dbclient sqlfile disk	SQL	Access to 'master' tables		
Exchange Exchangeserver exchangesite exchangeroutinggroup exchangeconnector exchangelink exchangequeue	WMI			
Citrix citrixserver citrixfarm citrixsession citrixclient	SNMP	Get Read Only		

MQ mqqueuemanager mqcluster mqrepository mqxmitq mqqueuelocal mqqueueremote mqaliasq mqqueue mqalias mqchsdr mqchsvr mqchannel mqchannelof mqchrqstr mqchclntconn mqchclusrcvr mqchclusdr webspheremq	Telnet NETBIOS	UNIX OS Account on MQ server that allows SUDO access to clusqmgr, dspmq, runmqtsr		
Weblogic j2eeserver jmsprovider jdbcprovider jdbcdatasource j2eeapplication ejbmodule webapplication servlet ejbstateless ejbstateful ejbentity ejbmessagedriven	JMX	JMX Mbean Server User/Password with Monitor role		
JBOSS jmsdestination jmsserver ejbcomponent webapplication servlet connectionpool j2eecluster	JMX	JMX Mbean Server User/Password with Monitor role		
SAP SAP server SAP site SAPService SAP support package SAP component	Application access	BAPI SAP User/password for SAPGUI		

Probe Installation

Agentless discovery implements a software-based probe to execute protocol discovery patterns. The number and location of probes depends on three factors:

- ❖ Firewall location – If applications that need to be discovered exist in a DMZ, then a separate probe is needed or the firewall can be penetrated. However, this is not a recommended practice.

- ❖ Geography or WAN – although discovery can run over a WAN, it may be desirable to place probes at major sites, DR sites, etc.
- ❖ Load – probes are memory intense, and although it is not common, multiple probes may be desirable for very large environments

Services Activation

After the required protocols and management services have been selected, the network, system administrators, DBAs and application managers should verify that the services are active, functioning properly and reachable. There are always some that need installation or upgrading. Corrupt CIMs are common.

The reason for this step is that a) the discovery won't return accurate results and b) it is normal for management services like WMI to be loosely or poorly maintained. Management services are used by a small IT community and people outside of operations, including Security, often are not familiar with the software.

This is the time to add CIs and/or attributes to services like SNMP. Management services support a rich set of data, although the default installs may be quite sparse. This capability is preferable to trying to federate or derive from external sources.

Initial Discovery and Baselining

Before activating discovery, inform the Security and Network groups ahead of time. This will allow them to deal with false virus alerts and other protective measures that have heuristics that trigger on unusual activities.

When the discovery results are available, organize the results so they can be compared to existing data sources. Possible sources are existing manual server lists, monitored nodes from your infrastructure/application monitoring tool, change management records, network reports, DNS listings and asset management reports. At a minimum, network nodes and servers should be reconciled. It is normal to encounter differences, but this usually results only in the adjustment of the IP ranges being discovered and the correction of misbehaving services.

Custom Discoveries

A discovery engine should provide the ability to extend and handle custom CIs. Custom here means enhancing the out-of-the-box discovery logic or patterns. Configuring custom discovery is a large, separate and vendor-specific topic.

There are three reasons for custom discovery:

- ❖ The existence of special devices like hardware-based routing/translation engine, SAN, etc.
- ❖ The existence of key products that are not universally adopted to the point that vendors discover them out of the box, for example, Tibco, Webmethods, ERP packages
- ❖ A need to reach deeper into a CIM or registry

Occasionally there is a need to instrument some code to get custom CIs. JMX provides a small set of methods that can be interfaced to, but it may be necessary to build a bean to get at the data.

Scheduling

A discovery protocol does not need to continuously run. Protocols or layers can be scheduled. Consider the following in selecting the schedule frequency:

- ❖ How often do they normally change, e.g., IP assignment are relatively stable?
- ❖ What is the potential for unauthorized changes?
- ❖ How frequently will the results be viewed?
- ❖ What is the risk and number of dependent CIs?

In addition to scheduled discovery, the CMDB Administrator will also need to run ad hoc rediscoveries for critical analysis or batch of major changes.

Probe Monitoring

The discovery probe is a critical infrastructure item and should be monitored by enterprise console like any key application. Send alerts to the CMDB Administrator or user if it fails to update for over 'n' hours past the normal scheduled time.

6. CI Ownership and Duties

A CMDB is an enterprise tool to coordinate and improve decision making across groups. Controlling configuration integrity is a shared responsibility and can not be assigned to a single person.

Formal CI Assignment to Domain Owners

Every person who is responsible for an operational component, whether it is hardware or software, is a CMDB end user, also referred to here as CI Domain Owners. The process starts by allocating ownership of each CI to one, and only one, individual. Viewing each other's CIs is fine, but one person has to be held accountable for how they are configured and changed. This minimizes the miscommunications and errors made while inserting changes. Ownership will vary by organization but the existing responsibilities will naturally dictate how ownership should be assigned.

In addition to ownership, each owner is always responsible for monitoring its neighboring CIs (in a configuration sense). A neighbor can be a peer CI on the same view level or higher or lower levels.

Outsourced entities, external providers and support groups should also be considered CI Domain Owners if they are part of the defacto operations. There is no reason to exclude vendors/providers, although viewing roles may be more tightly restricted.

Recurring Duties

Each CI Domain Owner has a set of views that they need to regularly review and consult. They also have the following configuration-related responsibilities:

- ❖ Review daily or respond to configuration events and notifications.
- ❖ Be aware of their neighbors and proactively communicate potential problems that they might induce in another's domain.
- ❖ Perform change impact analysis when a proposed RFC affects their domain.
- ❖ If they have replicated items, run regular delta analysis.
- ❖ Control drift from gold standard.
- ❖ Continually modify CIs and views to avoid repeat problems and change errors that can also include proposing new shared views that are shared.
- ❖ Adjust CIs and views to conform to vendor installation and configuration recommendations.
- ❖ Update how future event notifications are generated where more timely response can prevent problems.
- ❖ Maintain snapshots.

Continuous Improvement

The Change Manager, CAB, Architect, Operations and Application managers also have a role in reviewing corrective action from problem resolution, test failures and change post-implementation reviews. This role should question the accounting and viewing of CIs in relation to potential errors.

7. Relationship Mapping

Mapping relationships between CIs is the most important feature. Discovery finds the "dots" while mapping "connects the dots". Relationships create insights around where and how CIs affect each other. Without it, a CMDB is just another asset and inventory database.

Relationships are dependencies that affect how well another CI can function. A CMDB finds relationships using three processes.

- ❖ Inference through the observation of access and shared resources
- ❖ Direct input as a result of reading internal tables
- ❖ Enrichment

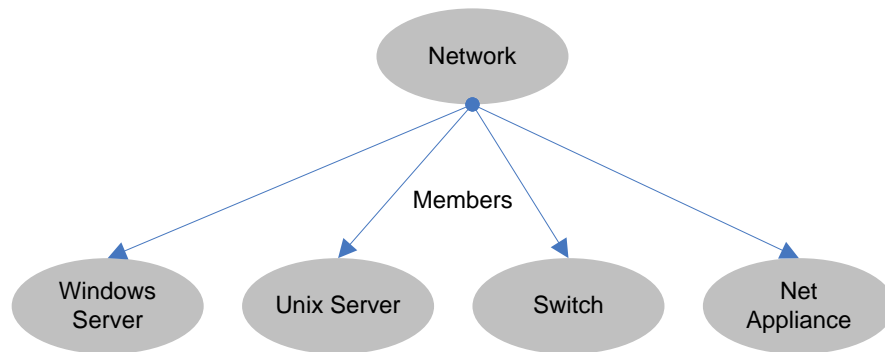
Understand Configuration Relationships

Relationships in a CMDB are unique to Configuration Management. No industry standard approach exists so it's important to understand how the vendor has implemented this function. Be aware that out-of-the-box mapping assumes certain routine conditions. For example, there are defacto port assignments; however, these can be changed. If, for example, you are not using port number 1521 for Oracle TNS Listener then mapping parameters will have to be adjusted.

Some common configuration relationships are:

- ❖ Contained or Is Contained By
- ❖ Member
- ❖ Uses
- ❖ Manages
- ❖ Etc.

More flexible CMDBs allow the definition of custom relationships. Relationships include not only the type but cardinality (e.g., locate all servers with two or more CPUs) and direction (e.g., CI being influenced or is the most dependent). Additionally Boolean logic may also be used on other properties to further define uniqueness.



Virtual Relationships

Virtual links are a special type of relationship. They can be used to account for virtualized servers, business functions, SANs and VLANs. These are vendor specific and implemented in later phases.

Define Mapping Scope

During the CI Needs Analysis a list of CI Types was defined. Some of these CIs can be viewed as a series of layers that begin with physical network up to business services. Example include:

- ❖ Physical Hardware
- ❖ Supporting Software
- ❖ Application Software
- ❖ Transactions
- ❖ Business Services or Processes
- ❖ People

Other CIs don't form obvious hierarchies like:

- ❖ Configuration Settings
- ❖ Rules
- ❖ Documents

All of the possible mapping relationship combinations need to be considered. Many will be automatically detected by the CMDB by inference or directly reading configuration-type files. Networks can be discovered using a combination of SNMP, IP sweep, Telnet, scan routing and switching tables. Where there is a gap, then an enrichment rule or discovery pattern must be created.

It is not necessary to exhaustively detect every relationship in the early phases. Managing a CMDB involves learning from daily operational errors and oversights by capturing additional dependencies after detrimental side effects are better understood. Allow time for the organization to mature its CMDB before adding complex correlation rules. Start with the simple rules: use and understand them before modifying them.

Enrichment

Discovery will find most but not all of the information in the CMDB. A key success factor is keeping manual data entry to the absolute minimum. Enrichment helps fill the gap. Enrichment is the step that uses a unique condition or pattern that is always true to create additional attributes by inference.

Many terms used by IT are arbitrary and identify logical entities such as Application for one. A basic definition might be a set of software that enables a business process. But what actually constitutes an application? Are forms and reports a set of executables? Does it include batch jobs, scripts, struts, help files, enterprise beans and source code? Point having been made, after discovery this analysis and configuration is required. Again each vendor handles this differently.

There is a question of how far to go with enrichment. The use cases defined earlier will bound how much mapping is done. A truly complete impact analysis capability needs to map from routers to end users. The amount of effort and how well higher levels of data are structured will affect what is feasible. At a minimum a complete mapping of applications is advisable, using databases and other key names, things and classifications that are used to communicate within IT daily.

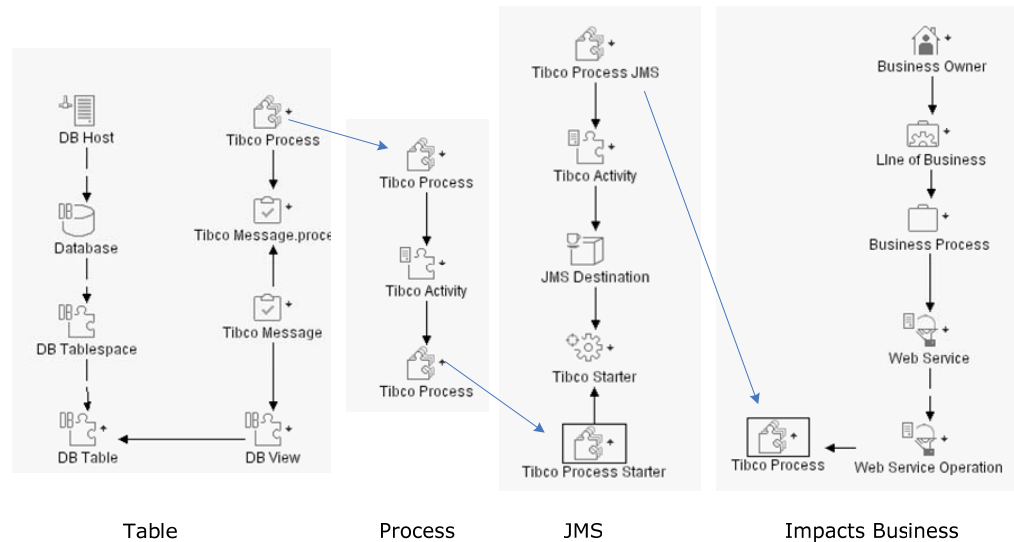
Correlation Severity

Understanding that a dependency exists does not complete the relationship mapping phase. There is also a need to know how critical the dependency is.

A simple correlation example is a Web Server Virtual Directory scenario. A Web Server uses multiple virtual directories located on separate file servers. If one of those file servers becomes unavailable, the Web server itself is not impacted, nor are any users of any other files on any of the unaffected file servers. But from the perspective of the user who needs files on the unavailable file server, there is an impact.

A related example might be when web clients are critically impacted but a NETBIOS client is not.

A more complex example follows that shows how a misbehaving Tibco process affects the business user through four layers.



The first step is assigning definitions for correlation ranges. Generic values can be used, such as Normal, Warning, Minor, Major and Critical, similar to application monitoring. A more useful categorization maps definitions to these values, such as:

- ❖ Prevents another CI from loading
- ❖ Performance degradation
- ❖ Induce instability
- ❖ Data corruption
- ❖ Security policy violation
- ❖ Overwrites key configuration files
- ❖ And many others

After defining the severity levels and obtaining some experience there are other refinements that could be made to the correlation:

- ❖ Should the degrees of separation or distance between CIs further weight the correlation severity?
- ❖ Should child severities propagate up to parent or peer CIs using a worse child or best child rule?
- ❖ Should time of day, current load, time in production or other factors modify the severity?

Finally, ensure that correlations are manually tested and ensure that the ongoing Change Control process includes checks to update the CMDB.

8. View Creation

Now that the CIs have been discovered and relationships are mapped, user views and reports can be configured. Like any reporting function, well-targeted and designed views are the key to better decision making.

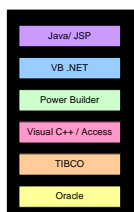
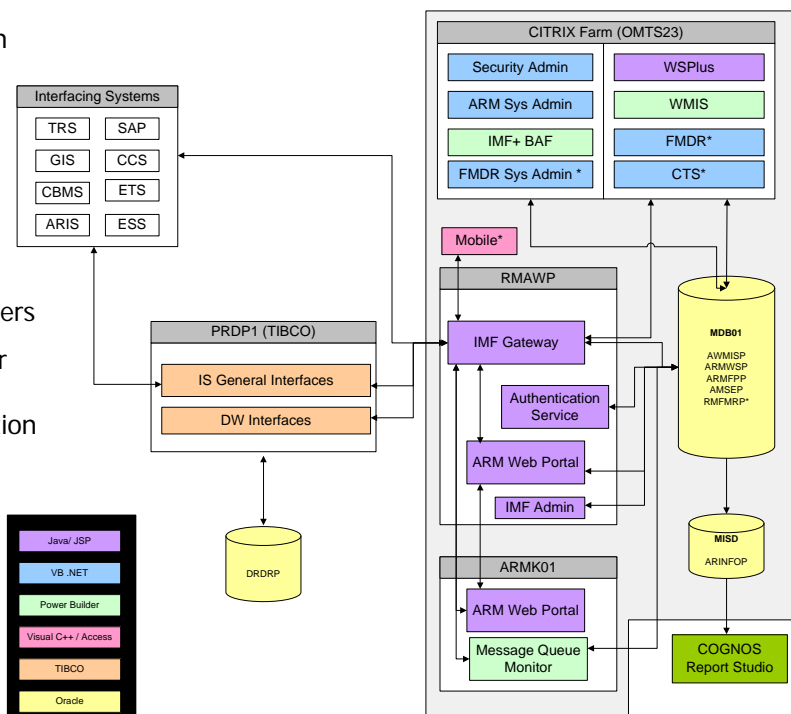
Views are displayed as maps and tabular reports. By far, maps communicate more information. Vendors provide widely differing visualization capabilities so this feature should be carefully examined before purchasing a tool. Navigate views hierarchically to higher and lower levels of detail, sideways to peers structures or neighbors and through the database when a view may be artificially limited by default but a full view of all components is necessary.

In addition to ad hoc queries, views are targeted to various stakeholders:

- ❖ General use
- ❖ Infrastructure domain owners, e.g., Network, System administrators, DBA, Ops
- ❖ Process owners
- ❖ Application owners
- ❖ Business system owners
- ❖ Outsourced and other external entities that participate in production operations

Views are also targeted to different purposes:

- ❖ Logical representation of IT assets in the organization
- ❖ To show conformance to policies, installation guides or other technical specifications
- ❖ Software bill of materials
- ❖ Built from the repository to represent relationships between components
- ❖ Examples of layered views for total topology or subsets like network, application or batch jobs



View Examples

An organization can accumulate hundreds of views. In section 2, Use Cases were defined. These should now prioritize your time in generating views. Here are some sample view scenarios.

A server is exhibiting a problem. An ad hoc query is configured to show what has changed to diagnose the disruption. A hierarchical view might be used in the following manner:

1. A change on a server A01 is observed.
2. Drilling in we reveals that the disk has changed.
3. An additional drill reveals which volumes have changed.
4. A subsequent drill reveals that the disk utilization is steadily climbing over time.

Drill down on all the software installed on one of the NT Servers. Events may also be added to show when future changes are detected, such as new versions or the removal of software from the machine.

Explode all items discovered on one Windows server to see installed services, software, IP addresses and socket connections that are open to other servers. When integrated to underlying Enterprise Management Systems such as OpenView and PATROL, key stats or alerts can be mapped to the infrastructure that might be added to what was discovered automatically.

To look for possible side effects, view all connections in and out of the multiple servers in supporting an application. There will be many HTTP connections in addition to other recognized sockets such as telnet, PATROL, NETBIOS and Oracle connections. Unrecognized connections are identified by their port number. Next drill down to view all connections going into the UNIX server. Most connections will be well-known but there are others (such as port 774 and 199) that are unidentified and suspect.

View Training

Data extraction and visualization typically does not use routine SQL but Query By Example or models. Each vendor implements views differently so training is required. Usually only the core project team is trained in views. Training is more effective if conducted on site so real examples can be used to communicate configuration concepts. Vendor capabilities vary widely so details in that arena will not be covered here.

View Plan

Eventually creating views will become a routine part of IT life. At the beginning it will help to establish a tentative implementation to help manage expectations.

Use Case	Question(s) Answered	Primary Stakeholder	Reusability

In addition to User Cases also consider risk, key applications and heavily used shared services. An example priority might look like:

View Category	Example
Topologies by layer	Network, Server farms, database
Application Views	Executables by servers Software classes User view - # accounts, max concurrent over some period, sessions Key errors thrown for critical components Open Changes Open Calls and Incidents
Shared Views	Asset view – license key expiration, ownership, lease, maintenance, contract Scheduled tasks
Ad Hoc	Are all Oracle instances currently dual homed
Process views	Service desk view: Escalations from 1-2 and 2-3, open incidents with high priority, calls over last 24 hours, incidents with multiple calls
Compliance	Forbidden software is not installed Vendor (or internally developed) software pre-reqs
Data flows	What participates in an add new account transaction
Change Impact	What is impacted and should be retested if we modify web service xyz
Comparisons	Do all of the web servers have the same hot fixes
Moves, reconfigurations	Are all the servers configured the same way after changing the subnet

Snapshots

Snapshots allow versions of a view and make later comparisons available for potential future use. Snapshots tend to consume storage rapidly so instruct users to snapshot judiciously. Three types of comparisons are used:

- ❖ Compound Comparison allows live comparison of two configuration items (CIs) and highlights the differences.
- ❖ Gold Standard Comparison lets you compare between a gold standard and live configuration items (CIs).
- ❖ Snapshot Comparison allows two configuration/application baselines taken at different points to illustrate changes over time.

Events

Adding configuration events and notifications complements views. A CMDB should continuously apply rules to new discoveries and raise events to the Domain Owners. The type of event is specific to the objects being monitored. This allows proactive management by alerting that condition should be reviewed.

Events might trigger an alert when (1) a key component has disappeared or (2) a new version has arrived or (3) when prohibited items are found or a configuration

neighbor has made a change. Event management is an advanced topic and can be implemented in the second or third phase of a CMDB.

9. CMDB Administration Processes

A CMDB like any other database application depends on effective management of certain ongoing processes. These processes do not manage CI content that have already been assigned to Domain Owners. They are different from business database applications, so special attention is required to ensure they are assigned and implemented. The identification and description of these processes is listed in the table below, but refining and formalizing these will need to be included in the particular implementation.

A CMDB Administrator will need to be assigned the responsibility for these processes, except for the DBA task, which should be handled as part of the normal database environment. After the initial installation it should take no more than two days a week. Good candidates for this role are Change Manager or a person who works with the technical architect or application lead or software configuration manager, but only if they are familiar with technical infrastructure. Operations personnel are not good candidates as they are unfamiliar with higher-level applications and business service CIs and relationships.

Admin Process	Owner	Trigger
Physical Maintenance	DBA	Daily monitoring
CI Type Change Control	CBDM Admin and CI Domain Owners	New procurement Change PIR (Post Implementation Review) Problem analysis
Discovery Monitoring	CBDM Admin	Daily monitoring
Global Changes	CBDM Admin or Architect	At direction of management
User Access	CBDM Admin or Security	User add and changes
Enterprise Views	CBDM Admin	User request
Key Data Design	CBDM Admin	Initial implementation or major CMDB changes

Physical Maintenance

The DBA will need to assist with monitoring the physical CMDB instance and table spaces. Although there may be millions of CIs the database should not significantly grow over time unless there is:

- ❖ Heavy use of snapshots, in which case it will grow quite rapidly
- ❖ Extend CIs with documents or CLOBs (which is not recommended)

Accepted practices for business applications such as running fine-grained daily backups, redundancy, maintaining test platforms and data, etc., are not critical for a CMDB unless a data center move or disaster recovery plan is in the offing. The reason is that if the database is lost, the discovery engine will rebuild almost all of the data. The key is to understand that it is just a tool and not mission-critical. If resources are limited, then save the storage space for snapshots.

CI Type Change Control

- ❖ As opposed to CI itself
- ❖ Secondary – need should come from Domain Owner
- ❖ Add/change CI Types and attributes
- ❖ CI fix (missing, duplicate attributes)
- ❖ Change primary key identifier
- ❖ Correction of CI assignment – can't be high level
- ❖ How would configuration information help?

Discovery Monitoring

The discovery engine(s) are scheduled processes. A monitor/alert needs to be configured to indicate failures or the discovery logs need to be inspected daily. Pay particular attention to monitoring memory and heap usage.

Some discovery engines allow the scheduling of runs by protocol or CI Type. Tradeoffs must be made to balance timeliness with execution time and change volatility. An example of this is the IP address changes which happen infrequently, making rediscovery a lower priority.

If your discovery engine utilizes credentials, then a process needs to be established with Security Administration and possibly the network, application and server administration groups if access permissions are managed at different points. This is different from CMDB user administration. Discovery credentials are usually in read-only administrator or system monitoring domain accounts. Credentials like a SNMP community string do not frequently change; however, a coordination point is still needed.

User Access

Users need to be granted role-based access to use functions within the CMDB. CI Domain Owners will have more permissions than those only viewing shared maps. Some network information is sensitive data that could provide damaging insight into the technical environment. Typically the user base for a CMDB is less than 75 and does not frequently change, so it should not require more than one hour a week.

Enterprise Views

Graphical maps, reports and other standard queries can be a bit different for a CMDB. The CMDB Administrator needs to be a go-to resource for configuring complex or enterprise views. Domain Owners need to be able to modify and maintain their own views.

Global Changes

The IT organization may occasionally implement large-scale changes in the infrastructure that affect many identifiers, IP DNS or how management services like JMX or SNMP are configured. The CMDB Administrator should manage the rediscovery process and report the changes so redundant events and deltas are understood.

Enhancement to Satisfy View Requirements

After CI Domain Owners begin using the initial set of views they will have new ongoing data requirements like any reporting system. Some views will require new CIs. The CMDB Administrator will need to activate and test new discovery packages or configure custom discoveries to meet future analysis needs.

Appendix 1 - Roles and Responsibilities

The following table describes the CMDB implementation team members for a very large enterprise. Smaller projects can scale down accordingly.

Project Role	Project Responsibilities	Ongoing Responsibilities
Sponsor/Stakeholder	<p>Defines Use Cases/Value realization, ultimately responsible for delivering ROI</p> <p>Owns Financial and contractual issues, non-technical aspects of CMDB</p> <p>"Sells" value throughout the organization, acts as a CMDB "Champion"</p> <p>Communicates goals, timelines, deadlines to Project Team and other Roles</p>	
CMDB Administrator	<p>Installs CMDB components, CMDB GUI Client, patches, updates, and upgrades</p> <p>Owns packages and physical parts of discovery patterns</p> <p>Maintains architecture and deployment documentation from PSO and afterward</p> <p>Owns stability and performance SLAs, implements best practices</p> <p>Owns system settings, limits, configuration, circuit breaker settings,</p> <p>Maintains events and event configuration</p> <p>Owns Backups and DR readiness plan and operations</p> <p>Connectivity and communication between server, gateway, probe, and client</p> <p>Owns and delegates authority to use the JMX administrative console functions</p> <p>Administers security, users, profiles, permissions, etc.</p> <p>Creates and deploys enrichments at the system level</p>	
Discovery Manager	<p>Owns logical parts of discovery patterns (function, parameters, etc.)</p> <p>Owns domain scope (IP ranges, Protocols, Credentials)</p> <p>Schedules discovery</p> <p>Runs on demand discovery or delegates authority to do so</p> <p>Validates discovery results, iterates rediscovery, repair discovery</p> <p>Works with Mapping specialist to gather discovery credentials</p> <p>Goes between CMDB administrator (which may be self in small shops) and the owners of the discovered infrastructure</p> <p>Ensures the discovery strategy supports the use cases for CMDB.</p> <p>Deploys/Maintains integration patterns to external applications</p> <p>Owns incoming data into the CMDB (manual entry, integration, etc.)</p> <p>Works with or IT the CMDB Configuration Manager</p>	

Mapping Manager	<p>Responsible for delivering value of Use Cases</p> <p>Interacts with Application SMEs, Owners, and Users</p> <p>Architects/implements Application Mapping strategies</p> <p>Architects/implements Reporting structures and usage, Change detection/management strategies, Correlation, etc.</p> <p>Architects/implements enrichments at the application level</p> <p>Goes between CMDB administrator (which may be self in small shops) and the owners of the discovered infrastructure</p> <p>Ensures the discovery strategy supports the use cases for CMDB.</p> <p>Works with CMDB Discovery Manager and Administrator to resolve issues</p> <p>Responsible for reporting value realization to Management / Stakeholders</p> <p>"Sells" value throughout the organization, acts as a CMDB "Champion"</p>	
DBA	<p>Creates database</p> <p>Allocates database space</p> <p>Performs security actions to allow CMDB installation</p> <p>Available for problem-solving during installation</p>	<p>Manages database performance and growth</p> <p>monitoring, tuning, reorgs, etc.</p> <p>Backups</p> <p>Allocates additional space as needed</p> <p>Available for infrequent problem-solving</p>
Server Administration	<p>Assists with scoping/hardware planning</p> <p>Procures/allocates CMDB component hardware</p> <p>Verify OS/software compliance</p>	<p>Manages CMDB server hardware</p> <p>Manages CMDB server/OS (space, patches, etc.)</p>
Network Administrator	<p>Verifies network connectivity</p> <p>Assists in probe deployment planning</p>	<p>Maintains component connectivity</p> <p>Maintains connectivity to target environments</p>
Security Administrator	<p>Facilitates credential-gathering Processes for installation</p> <p>Facilitates credential-gathering Processes for discovery</p> <p>Configures firewalls (may be Network Admin)</p> <p>Verifies CMDB security precautions</p>	<p>Responsible for CMDB Component security</p> <p>Controls/delegates discovery credentials</p>
Application SME	<p>Consults with application mapping specialist</p> <p>Verifies use cases</p> <p>Provides initial documentation of existing, manually assembled maps (e.g., Visio diagrams)</p>	<p>Validates application maps</p> <p>Identifies superfluous/missing components</p> <p>Possibly, the end-users of the mapping effort</p>
Users / End beneficiaries of the project	<p>Use cases verified</p> <p>Demo/training</p>	<p>Validates use case implementation</p> <p>Provide qualitative feedback</p>

Appendix 2 – Skill Set

Three of the project roles described in Appendix 1 need certain skills ideally. If possible assign individuals with the following backgrounds.

CMDB Administrator

The CMDB Administrator is a technical IT position, requiring a standard combination of server, networking, database and application support skills. The focus with the CMDB Administrator is maintaining performance and availability of the CMDB server and ensuring the CMDB architecture is deployed as designed and documented as required. The major skill levels and areas required for the CMDB Administrator are:

- ❖ Networking:
 - Basic understanding of TCP/IP networks and network hardware
 - Firewalls and DMZs
 - Basic Connectivity concepts such as networking protocols and packet sniffers
- ❖ Windows:
 - Basic Windows administration skills
 - Windows domains
 - NETBIOS/drive mapping/windows networking and connectivity
- ❖ UNIX:
 - Basic understanding of UNIX OS and concepts:
 - UNIX shells and scripting
 - UNIX file systems
 - UNIX networking concepts
- ❖ Database:
 - Basic MSSQL or Oracle user skills
 - Basic SQL knowledge, SQLPlus for Oracle, Enterprise Manager for MSSQL
 - Basic understanding of basic database architecture, e.g., table spaces, allocated disk space, etc.
- ❖ J2EE: Basic understanding of java and J2EE application architectures
- ❖ Application:
 - Ability to manage and support a distributed, enterprise-level application
 - Ability to diagnose/troubleshoot problems
 - Experience working with vendors and vendor support
 - Ability to trace transaction flow and use logging facilities for problem solving

CMDB Discovery Manager

The CMDB Discovery Manager is a technical IT position, requiring a broad background in multiple IT disciplines. A competent discovery manager is key to a successful implementation. A wise customer will choose their most technically competent resource available for this position. The major skill levels and areas required for the CMDB Discovery Manager are:

- ❖ Networking:
 - Advanced understanding of TCP/IP networks and protocols
 - Good understanding of network hardware and network devices such as Firewalls, DMZs, load balancers
 - Advanced connectivity concepts such as networking protocols and packet sniffers, VLANs, bandwidth
 - Networking protocols:
 - SNMP
 - WMI
 - UNIX and Windows Command Line protocols (TTY/SSH/Telnet)
 - JMX
 - Application-specific protocols (generic)
- ❖ Windows:
 - Advanced Windows administration skills, including WMI
 - Windows domains
 - NETBIOS/drive mapping/windows networking and connectivity
- ❖ UNIX:
 - Advanced understanding of UNIX OS and concepts:
 - UNIX shells and scripting
 - UNIX file systems
 - UNIX networking concepts
 - UNIX administration commands used for discovery, such as ifconfig/ipconfig, df, ps, etc.
- ❖ Database:
 - Basic MSSQL or Oracle user skills, ability to interpret system-level SQL statements used for discovery
 - SQL knowledge, SQLPlus for Oracle, Enterprise Manager for MSSQL
 - Basic understanding of basic database architecture, e.g., table spaces, allocated disk space, etc., discovered database resources
- ❖ J2EE:
 - Basic understanding of java and J2EE application architectures, EJB, administration consoles, J2EE servers and domains and JDBC

- ❖ Application:
 - Good understanding of older as well as modern application architectures; 2- and 3-tier applications, OSI model of layers
 - Ability to architect and deploy discovery processes:
 - Discovery scheduling
 - Working with other IT administration groups such as networking, DBAs, Server Administrators and Security Administrators
 - Experience working with vendors and vendor support
 - Ability to trace transaction flow and use logging facilities for problem solving
 - Ability to effectively communicate discovery (CMDB population) results and capabilities to management

Mapping Manager

The CMDB Mapping Manager is a technical IT position, but less technical and more applications- and business-focused than the other two CMDB areas. The major skill levels and areas required for the CMDB Discovery Manager are:

- ❖ Business:
 - Understanding of internal organization structure, LOBs
 - Intimate knowledge of organization's management structure, formal and informal political processes, lateral flow of knowledge and power within and between business units
 - How to find and use specific resources within the organization
 - Ability to create a structure and execution process for application mapping that complies with and is effective within their company
 - Ability to manage a team of application mapping specialists
- ❖ Presentation and Communications:
 - Ability to work with management and technical people of all management levels and skill sets
 - Persuasive speaking, ability to "sell" CMDB throughout the organization
 - Ability to present application mapping requirements and results to Application and IT Management
- ❖ Application:
 - Intimate understanding of the company's business services, applications and IT infrastructure
 - Good understanding of older as well as modern application architectures; 2- and 3- tier applications, OSI model of layers

- Basic Understanding of application tiers, platforms and up/down stream dependencies
- Ability to interface with application SMEs: to communicate requirements, understand answers and interpret those answers into the appropriate application mapping tasks

Appendix 3 – Current CMDB Vendors

For reference purposes, here are the current CMDB players in alphabetic order.

- ❖ BMC Atrium, Marimba
- ❖ Computer Associates purchased from Cendura
- ❖ EMC Smarts Application Discovery Manager purchased from nLayers
- ❖ HP uCMDB purchased from Mercury who purchased Appilog
- ❖ IBM CCDM purchased from Collation
- ❖ Microsoft, a neophyte CMDB in their Service Center product
- ❖ Tideway
- ❖ Symantec purchased from Relicore

Product Weaknesses to Avoid

- ❖ Agent-based technology on Solaris, AIX or Windows 2K only or for limited versions)
- ❖ No ability to map network devices
- ❖ Cannot map DB, .Net or J2EE application components
- ❖ No ability to detect changes to Configuration Parameters stored in DB
- ❖ No ability to detect application intra-dependencies
- ❖ Immature product
- ❖ Discovers only J2EE App Servers
- ❖ Limited Network service discovery (LDAP, NFS & DNS)
- ❖ CMDB is mainly a SDK and each installation is a custom development
- ❖ No discovery engine
- ❖ Rely on third-party events to build topology
- ❖ Only supports Oracle or MS SQL as database
- ❖ Multiple products required to achieve complete Application Mapping Solution
- ❖ Application discovery based on network traffic fingerprinting
- ❖ Combination of non-integrated products
- ❖ No integration services
- ❖ Poor visualization and mapping
- ❖ No event management, alerting or notifications
- ❖ Can't add custom discovery patterns

About Evergreen Systems

Evergreen Systems is a highly specialized technology consulting firm focused on helping companies maximize the value of IT to their businesses. Evergreen specializes in ITIL (Information Technology Infrastructure Library) business case development and consulting and offers services that include Service Catalog development, Change Management consulting, Configuration Management Database (CMDB) development and Asset Management. From strategic planning, to policy development, through execution, Evergreen makes sure that what gets planned, gets done.

Leaders in insurance, finance, healthcare and retail rely on Evergreen to address today's major business challenges, including making ITIL and COBIT operable; understanding and organizing IT assets for better planning and execution; developing automated, streamlined compliance processes; and maximizing benefits through change, configuration and asset management. Global 2000 organizations work with Evergreen for sound strategy, flawless execution and measurable results.