Une CMDB à l'Université de Strasbourg

Virgile Gerecke
Direction Informatique
14 rue René Descartes
67 000 Strasbourg

Julien Dupré
Direction Informatique
14 rue René Descartes
67 000 Strasbourg

Résumé

La CMDB, ou base de données de gestion des configurations, est au cœur du processus de gestion des configurations. C'est un des premiers et des rares outils mis en avant par ITIL. Plus qu'un inventaire, la CMDB est censée faire le lien entre différents éléments issus de multiples inventaires et référentiels et apporter une version historisée de ces éléments et de leurs relations. Pour autant, c'est un outil qui, s'il semble prometteur au sortir d'une formation ITIL, est surtout très théorique et difficile à implémenter. En 2009 un premier groupe de travail de la Direction Informatique de l'Université de Strasbourg avait d'ailleurs planché sur le sujet de la CMDB sans aboutir à une implémentation possible, fut-t-elle parcellaire. Enfin, la plus-value par rapport à un outil d'inventaire classique n'est pas évidente au premier abord mais se révèle incontournable à l'usage.

S'appuyant sur les démarches qualité entreprises depuis 2009, la Direction Informatique de l'Université de Strasbourg a lancé depuis 2013 un projet de mise en oeuvre d'une CMDB portant sur un vaste périmètre. Un catalogue d'une centaine de services offerts est finalement lié à plus de 300 applications, 200 serveurs physiques, 1800 équipements réseaux, 200 lieux, 300 "clients" et contacts, 400 organisations et structures. Les données évoquées représentent plus de 13 000 objets qui auront été injectés dans la CMDB. La réalisation de la CMDB s'appuie sur iTop, un produit dont les qualités et les limites seront exposées lors de la présentation. Nous nous attacherons particulièrement à démontrer la souplesse de l'outil et sa capacité à s'interfacer avec de nombreux éléments d'un système d'information de façon efficace.

Mots clefs

ITIL, CMDB, ITOP, Gestion des configurations, inventaire, données de référence, ETL

1 Introduction

1.1 Contexte

L'Université de Strasbourg a été créée en 2009 suite à la fusion de 4 établissements strasbourgeois préexistants. Du coté des services informatiques, 9 entités se sont réunies pour donner le jour à une direction des usages et une direction informatique (DI). Cette dernière structure, forte initialement de 122 personnels, a dû affronter le défi que représente la gestion d'un système d'information complexe et vaste en terme de nombre d'applications, ou d'éléments d'infrastructure. Un catalogue d'une centaine de services offerts est

finalement lié à plus de 300 applications, 200 serveurs physiques, 1800 équipements réseaux, 200 lieux, 300 "clients" et contacts, 400 organisations et structures. Les données évoquées représentent plus de 13 000 objets. Bon nombre d'entre-elles étaient préalablement référencées dans des inventaires comme *GLPI* pour les éléments d'infrastructures ou dans un wiki documentaire pour les services.

Le recensement et la cartographie même partielle de ces éléments n'ont pas été une mince affaire. Il a d'abord fallu homogénéiser les pratiques et trouver un langage commun au travers de la mise en place d'une démarche qualité centrée sur ITIL dès 2009. Cette démarche a fait l'objet d'une présentation lors des JRES en 2013. Elle a doté la DI, via le lancement de 11 projets, d'outils structurants comme des processus de gestion des incidents ou un catalogue des services. Ces outils ont "résisté" à 3 organisations et 4 directions différentes. Malgré des limites, ils ont apporté des gains qui ont assuré leur pérennité. Dans le cas de processus de faible valeur ajoutée, ils ont sombré dans un oubli bienheureux en attendant peut-être des jours meilleurs. Dans le même temps, un recensement des applications était en cours sous l'angle de l'architecture du système d'information. Enfin, de son côté, le support chargé de gérer plus de 25 000 tickets par an avait besoin de s'outiller en terme de procédures et de documentations pour pouvoir répondre à la demande. Un wiki a servi de déversoir dans lequel toutes ces informations étaient jetées avec plus ou moins de bonheur. Si le recensement et le travail de documentation étaient effectués, il est devenu extrêmement difficile de maintenir ces données. De plus les liens et corrélations à faire entre différents éléments se faisaient alors difficilement. Comment lier ces informations ? Comment en assurer la mise à jour ainsi que la cohérence ?

1.2 Historique du projet

Dès 2009, le concept de CMDB avait fait quelques émules et semblait être un Graal à atteindre. La CMDB (Configuration Management DataBase) doit référencer toutes les informations utiles à l'exploitation des services informatiques. Les éléments composant la CMDB sont appelés CI ou "Configuration Items". Il peut s'agir de services, d'applications, de serveurs ou d'équipements réseaux, de postes de travail, de documentations de contrats ou de procédures. La CMDB est donc censée référencer ces éléments et les lier entre eux. Elle doit également historiser les informations et c'est (outre son périmètre très large) ce qui la distingue d'un inventaire classique. Dès 2009, un premier groupe pilote avait planché sur la mise en place de cet outil merveilleux, sans aboutir. Le manque d'outils disponibles, l'ampleur de la tâche avait finalement découragé les plus enthousiastes.

En 2012, un peu de veille technologique et la comparaison de divers outils libres (*oneCMDB*, *iTop*) ont finalement conduit à la mise en place d'une maquette d'*iTop* un outil qui avait été présenté aux RMLLs à Strasbourg. *iTop* est un logiciel libre, développé principalement par la société Combodo qui se présente comme un outil ITSM et CMDB. Au-delà de la CMDB, la promesse d'iTop est donc d'aider à la gestion des services informatiques. Des processus intégrés à l'outil comme la gestion des changements ou la gestion des incidents (système de tickets) doivent donc apporter un plus, comme si la seule CMDB ne suffisait pas aux ambitions des développeurs. On peut aussi considérer qu'il s'agit au final d'utiliser les éléments de la CMDB afin de faciliter le travail quotidien. *iTop* est développé en PHP et est accessible via une interface web. Il possède deux qualités principales :

- des mécanismes d'import et de synchronisation qui feront l'objet de la première partie de cet article ;
- un modèle de donnée extensible qui sera présenté dans une seconde partie.

Malgré ces qualités, la maquette *iTop* a rapidement pris la poussière courant 2013. Personne ne semblait finalement s'y intéresser. En 2014, l'étude a été réactivée car de nouveaux besoins ont émergé qui pouvaient être satisfaits par la maquette. Ces moments sont instructifs et nous étudierons dans cet article l'outil plus en détail afin d'en présenter les qualités et les limites, les difficultés à se l'approprier. Cela fera l'objet d'une troisième partie qui sera également l'occasion de présenter les apports inattendus de la démarche.

Au final, si on peut s'interroger sur ce qui a pu motiver une telle démarche, il faut considérer la complexité et le périmètre des éléments gérés. La désagréable sensation de construire sur des fondations dont la nature serait inconnue ainsi que la volonté de décloisonner les informations ont conduit au lancement de ces expérimentations. Se lancer dans la CMDB, c'est essayer de répondre à l'injonction écrite sur le fronton du temple de Delphes : "connais-toi toi-même".

2 Synchronisations et imports

Une fois les premiers tests manuels effectués, vient rapidement la tentation d'insérer des données réelles dans la CMDB naissante. Le produit permet des imports *CSV* et propose des tables de synchronisation. On pioche, on agrège, on lie les éléments entre eux plus ou moins proprement. On obtient rapidement une plateforme de démonstration permettant d'évaluer le potentiel de la démarche. Mais tout se complique lorsqu'il s'agit de mettre en place les éléments nécessaires à une mise en oeuvre effective et au maintien de la pertinence des données. Voici donc les étapes suivies à l'Université de Strasbourg.

2.1 Identification des sources

La première étape d'intégration des données est l'identification des sources. Nous avons donc procédé à un "inventaire des inventaires" à notre disposition. Nous avons donc retenu :

- les inventaires d'infrastructure (*GLPI*) ;
- les listes des services et des applications (notre wiki) ;
- les contacts, organisations (dans le référentiel des personnes et des structures) ;
- les bâtiments (base métier référentiel immobilier) ;
- les fonctions (base métier RH et application de gestion des correspondants) ;
- les contacts et utilisateurs (annuaire LDAP) ;
- le périmètre d'intervention de la direction (fichier tableur) ;
- l'inventaire des onduleurs et des climatisations (fichier tableur) ;
- les listes des salles gérées avec les informations récoltées par la DI au fur et à mesure des années (fichier tableur).

Avec l'aide des experts des différents domaines couverts par la CMDB, nous avons réalisé un tableau récapitulatif de l'ensemble de ces données comprenant la description de la source, les champs à intégrer et les règles d'usage (par exemple les règles concernant les statuts des objets).

Cette étape a permis d'obtenir une vision globale sur les données à utiliser. Mais elle a aussi été l'occasion de procéder à l'analyse de notre capacité à maîtriser nos données dans le temps. En effet, bien avant la CMDB, des liens existaient entre les données de certaines bases. Malheureusement, ces liens pouvaient souffrir d'obsolescence d'une part à cause du type de lien et d'autre part à cause des données elles-mêmes. L'exemple le plus flagrant était le lien entre les lieux et les équipements. D'un côté, les lieux avaient été importés une fois dans l'outil d'inventaire d'infrastructure sans synchronisation ultérieure. En conséquence, la création ou la modification de salles entraînait la perte de cohérence. De l'autre côté, le référentiel des lieux montrait des décalages entre les données saisies et l'évolution de la finalité des salles. Typiquement, des salles contenant des matériels comme des postes pédagogiques et qui étaient donc des salles de TP informatique pouvaient être référencées dans l'inventaire des lieux comme des salles de réunion ou des locaux techniques. Le travail sur la CMDB a permis de mettre en lumière ces deux aspects et de les corriger. Ce travail fastidieux effectué, les données ont pu être intégrées dans *iTop* tout en nous rappelant l'importance des contrôles de cohérence des données dans le cycle de vie des objets.

2.2 ETL - *Extract Transform Load*

Le principe de synchronisation d'*iTop* repose sur ce paradigme simple

- on extrait depuis les bases sources à travers les interfaces d'exposition des données ;
- on transforme ensuite ces données afin qu'elles soient compatibles ;
- on charge les données dans la CMDB.

2.2.1 Elles sont fraîches mes données !

Nous avons donc utilisé un produit interne appelé *synchronisator*. Cet outil générique permet de récupérer des données dans une source spécifiée à l'aide de services web ou de requêtes *SQL*. Nous exploitons donc les possibilités d'accès aux données de nos inventaires et à défaut, nous avons construit une bibliothèque de requêtes *SQL*.

2.2.2 Elles sont belles mes données !

Les données sont ensuite transformées pour correspondre d'une part aux règles d'usage définies lors de l'identification des sources mais aussi pour consolider les liens entre les éléments d'inventaire. En effet, *iTop* utilise un système d'héritage pour les éléments de configuration. Cela permet de partager des propriétés communes entre les classes d'objet. Nous avons ainsi défini un identifiant extérieur permettant de réconcilier les données lors des synchronisations. Il faut donc s'assurer de l'unicité des identifiants externes des objets pour éviter les problèmes d'ambiguïté sur les liens. Car si *iTop* permet à deux éléments d'avoir un même identifiant externe, il provoque une erreur d'import en cas d'incertitude. Parmi les transformations de données, *synchronisator* applique des règles de construction d'identifiant unique en préfixant les identifiants externes.

Dans certains cas le modèle d'*iTop* nous impose l'utilisation de classes d'objet. Ainsi, nos sondes d'inventaire ne remontent pas de données concernant les hyperviseurs. Comme cette information sera peut-être utilisée dans le futur, nous n'avons pas souhaité altérer le modèle de données et nous devons donc créer des objets virtuels hyperviseurs afin de lier entre eux serveurs et machines virtuelles. *Synchronisator* permet de créer les éléments *"virtuels"* qui ne remontent pas automatiquement dans l'inventaire. Dans notre cas, il s'agit par exemple des hyperviseurs. *iTop* définit les machines virtuelles comme on le voit sur la figure 1 :

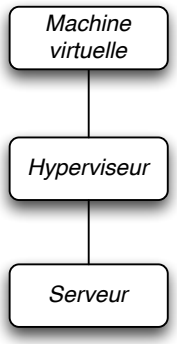

Figure 1 - Elément "virtuel" hyperviseur

2.2.3 Et ce sera tout ?

L'intégration des données se déroule en deux temps. Tout d'abord il faut remplir les tables de synchronisation d'*iTop* avec les données préparées. Dans notre cas, c'est une nouvelle fois *synchronisator* qui s'en charge mais d'autres scripts maisons peuvent faire l'affaire. Ensuite, les possibilités d'intégration et la flexibilité d'*iTop* entrent en oeuvre. En effet, lors de la définition d'une synchronisation, une table est créée avec tous les champs d'une classe *iTop*. Pour chacun de ces champs, on peut définir comme on le voit sur la figure 2 :

- si le champ participe à la réconciliation des données, avec la possibilité de réconcilier sur plusieurs champs ;
- si le champ doit être mis à jour ;
- la politique concernant la mise à jour des données (écrasement ou alimentation si vide) ;
- si le champ peut-être mis à jour manuellement ;
- la clé de recherche pour les objets liés (par exemple le nom de l'organisation dont dépend un élément de configuration, ce qu'on qualifie de jointure en *SQL*).

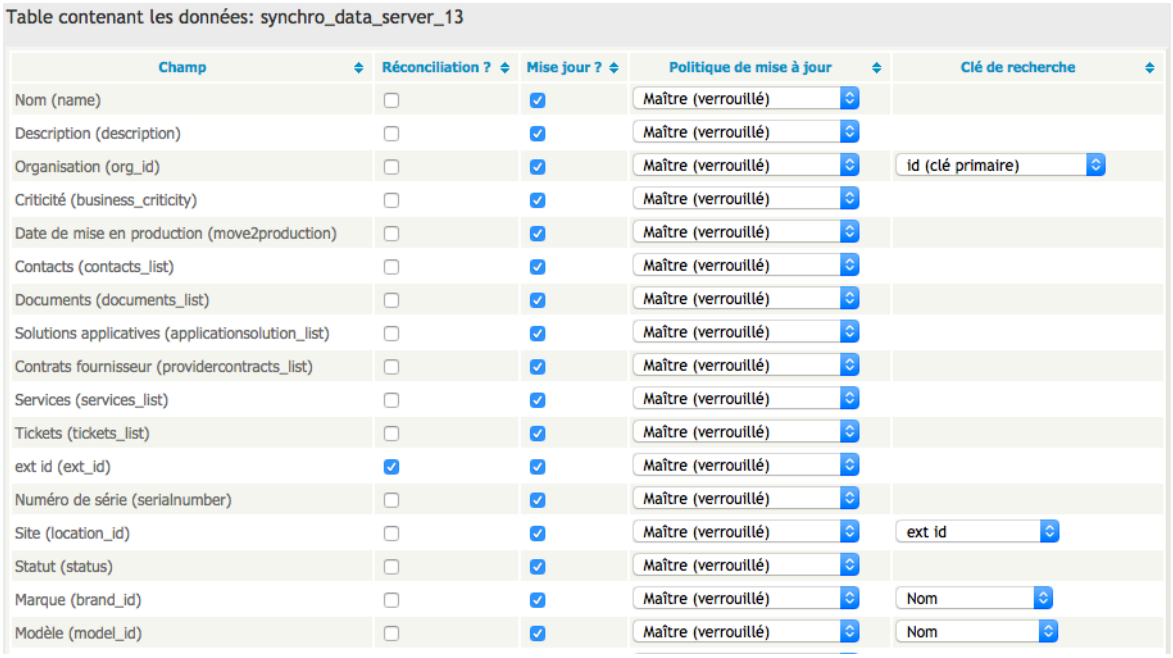
Table contenant les données: synchro_data_server_13

Champ	Réconciliation ?	Mise jour ?	Politique de mise à jour	Clé de recherche
Nom (name)	☐	☑	Maître (verrouillé)	
Description (description)	☐	☑	Maître (verrouillé)	
Organisation (org_id)	☐	☑	Maître (verrouillé)	id (clé primaire)
Criticité (business_criticity)	☐	☑	Maître (verrouillé)	
Date de mise en production (move2production)	☐	☑	Maître (verrouillé)	
Contacts (contacts_list)	☐	☑	Maître (verrouillé)	
Documents (documents_list)	☐	☑	Maître (verrouillé)	
Solutions applicatives (applicationsolution_list)	☐	☑	Maître (verrouillé)	
Contrats fournisseur (providercontracts_list)	☐	☑	Maître (verrouillé)	
Services (services_list)	☐	☑	Maître (verrouillé)	
Tickets (tickets_list)	☐	☑	Maître (verrouillé)	
ext id (ext_id)	☑	☑	Maître (verrouillé)	
Numéro de série (serialnumber)	☐	☑	Maître (verrouillé)	
Site (location_id)	☐	☑	Maître (verrouillé)	ext id
Statut (status)	☐	☑	Maître (verrouillé)	
Marque (brand_id)	☐	☑	Maître (verrouillé)	Nom
Modèle (model_id)	☐	☑	Maître (verrouillé)	Nom

Figure 2 - Configuration d'une synchronisation

Une fois les données présentes dans ces tables de synchronisation, *iTop* se charge de ventiler les informations dans son propre schéma de base de données. Ce mode de fonctionnement permet de conserver l'historique des modifications et de s'abstraire des subtilités du modèle de données, rendu complexe par le système d'héritage entre objets. Il permet aussi d'assurer la pérennité des synchronisations. Il faut noter qu'une attention particulière doit être portée aux clés externes permettant la réconciliation des données. Nous avons dû mettre en place, malheureusement après l'apparition de conflits, une nomenclature permettant d'exclure tout recouvrement sur ces clés externes.

L'ensemble des éléments nécessaires à la remontée automatique des informations depuis nos différents inventaires a nécessité environ un mois-homme.

2.3 Récupération manuelle de données

La démarche de mise en place d'une CMDB a aussi été l'occasion de référencer toutes les données, éléments de configuration ou liens entre éléments, stockées hors base de données normée. Il s'agissait essentiellement de fichiers tableurs mis en place en l'absence d'outil adéquat ou abordable pour la gestion de ces données. Les informations pouvaient être fragmentées (il existait plusieurs fichiers de gestion de salles) et les données constitutives n'étaient que faiblement liées au S.I. Nous avons essayé d'automatiser le plus de rapprochements possibles pour les données simples, en essayant d'interpréter les informations saisies comme le nom des contacts, tout en gérant les éventuelles fautes de frappes ou les différences de format. Pour le reste, un fastidieux travail manuel a été mené afin de réconcilier les éléments textuels avec les données du S.I. et ainsi exploiter la somme de connaissances accumulées dans ces fichiers.

Les scripts de rapprochement des données ont été réalisés en deux jours-homme. Le rapprochement manuel a représenté près d'une semaine-homme juste pour la récupération du fichier tableur liées aux données d'escalade. Toutes ces données ont ensuite été intégrées au format CSV dans *iTop*, grâce à un module dédié à ce type d'opération.

2.4 Cycle de vie des objets

2.4.1 iTop comme source de données

Notre CMDB agrège, intègre et lie les éléments de configuration provenant de sources diverses. Nous avons vu au paragraphe 2.3 qu'*iTop* servait également de source de données. En effet, les informations sont directement saisies et gérées dans l'outil dans le cas d'objets pour lesquels il n'existe pas de référentiel (ou pas de référentiel satisfaisant). C'est le cas de notre catalogue des services par exemple.

Toutefois, *iTop* permet aussi d'enrichir les données d'objets provenant de nos inventaires avec par exemple l'ajout de commentaires techniques propres à nos usages et qui n'auraient pas d'intérêt pour les utilisateurs initiaux. Ainsi on voit sur la figure 3 que le champ description que nous avons ajouté n'est pas verrouillé :

Figure 3 - Champs verrouillés vs saisie manuelle

Enfin, et il s'agit d'un des principaux intérêts de la démarche, des liens auparavant non référencés sont mis en place et donnent corps aux informations déjà présentes. C'est le cas par exemple des liens entre les instances de base de données et les applications qui les utilisent. Aucun référentiel ne nous donnait cette information qui semble pourtant utile dans le cadre de l'exploitation quotidienne des applications.

2.4.2 Suppression

Une fois qu'un objet est créé, il est lié à d'autres et peut même être enrichi. S'il provient d'une source externe, il a fait l'objet d'une synchronisation et des restrictions peuvent s'appliquer sur les manipulations dont il peut faire l'objet. Ainsi, certains champs alimentés automatiquement peuvent être verrouillés. C'est pourquoi *iTop* ne permet pas de suppression à partir de synchronisation. Cela évite les suppressions non désirées (lors d'une synchronisation qui échoue par exemple) et force à adopter une stratégie spécifique. Tous les objets que nous manipulons ont donc été pourvus d'un statut qui permet de gérer le cycle de vie. Ainsi, un serveur peut être en stock, puis avoir un usage défini et enfin être retiré. Il reste quand même présent dans la CMDB. Avant toute suppression, une analyse de la situation est menée afin de s'assurer que tous les liens vers ce serveur ont bien été mis à jour dans la réalité et dans l'outil. Une fois ces vérifications nécessaires effectuées, il peut être supprimé en toute sécurité des synchronisations et des objets gérés. Cela permet à la fois de limiter les risques de perte d'information côté CMDB mais aussi et surtout d'exploiter ces données afin d'éliminer tout effet de bord indésirable au retrait de l'objet.

Il est étonnant de constater que si *iTop* propose des statuts prêts à l'emploi pour plusieurs éléments de configuration (comme les serveurs), tous les objets ne peuvent être inactivés nativement (instances de base de données ou utilisateurs LDAP par exemple). Nous avons compensé ces manques par l'ajout des champs correspondants.

3 Modèle de données

3.1 Pourquoi étendre le modèle de données ?

Le modèle de données est l'élément central de la gestion des configurations. *iTop* propose un modèle par défaut riche et proche de nombreux usages. Nous avons vu qu'il était nécessaire de faire évoluer ce modèle car la CMDB est aussi source de données (cf. 2.4.1) et bien souvent des informations qui sont importantes dans notre contexte d'utilisation ne sont pas présentes dans l'outil par défaut.

Par exemple, pour notre catalogue de services, nous avons dû ajouter des champs qui n'étaient pas présents dans *iTop* comme la criticité du service, le fait qu'il soit ou non utilisé par certaines populations (étudiants, enseignants/chercheurs, personnels administratifs et techniques, etc.) ou le fait que certains services soient internes à la DSI.

Par ailleurs, toujours concernant la modélisation des services, nous aurions pu reproduire la structure existante réalisée à partir de notre wiki (figure 4).

Mais cette représentation ne reflétait ni la diversité de nos clients, ni les différents environnements mis en œuvre. Nous avons donc opté pour une modélisation plus ambitieuse et plus proche de la réalité (figure 5) qui permet d'exploiter au mieux les fonctionnalités d'étude d'impact et de dépendance de l'outil (voir 4.5) :

3.2 Comment étendre le modèle de données ?

Le modèle d'*iTop* est fait pour être étendu. Tous les objets sont décrits sous forme de classes qui bénéficient d'un système d'héritage. Il est donc aisé de créer de nouvelles classes, de modifier des classes existantes ou d'étendre des classes.

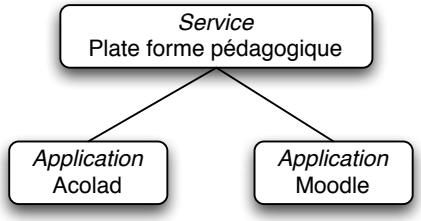

Figure 4 - Modélisation des services et applications dans le wiki

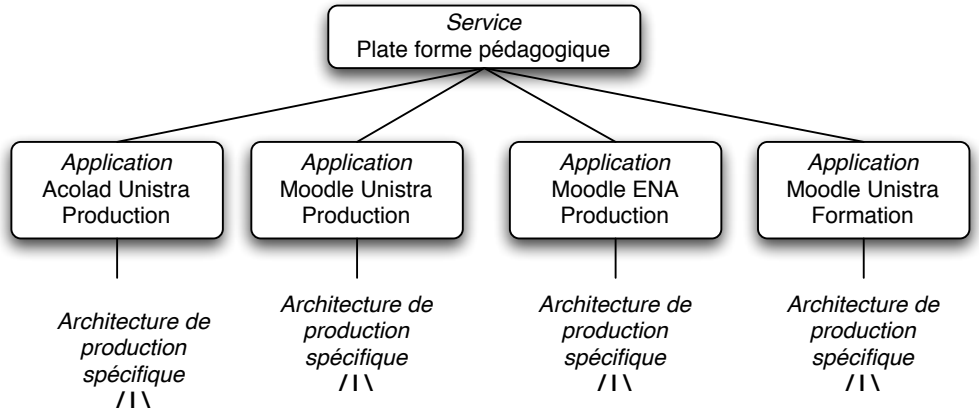

Figure 5 - Modélisation des services et applications dans iTop

L'ensemble de ces classes est décrit sous forme de fichiers *XML* de configuration comprenant :

- des données de configuration de stockage ;
- la liste des champs et de leurs données jointes (la plupart des éléments est liée à une organisation dont on stocke l'identifiant *org_id* mais dont on affiche simplement le nom grâce à une jointure implicite offerte par *iTop*) ;
- la configuration de l'affichage des données de la classe ;
- la configuration des formulaires de recherche d'objets.

iTop propose de nombreuses classes dans son modèle par défaut. Afin de préserver les possibilités de mise à jour, toutes les modifications du modèle passent par un système de modules. Nous pouvons donc créer des modules spécifiques à chaque but visé, comprenant une conjonction de classes modifiées et/ou nouvelles.

La mise en place d'une CMDB a créé de nombreuses attentes, portant sur diverses problématiques. Nous avons donc créé des modules pour :

- les fluides (onduleurs, alimentations, climatisations qui sont des notions absentes d'*iTop*) ;
- les salles (extension des lieux pour permettre d'apporter de nombreuses précisions) ;
- de nombreux liens avec des contacts (nos correspondants au sein des structures clients, des responsables administratifs ou de filière, etc.) ;
- l'architecture de nos applications comme décrite au paragraphe 3.1 ;
- les flux de données entre les différents éléments du S.I.
- etc.

Ce ne sont que nos premiers modules car nous continuons d'adapter le modèle à l'évolution de nos usages. Au-delà de l'aspect technique, nous avons aussi mis en place un processus d'évolution du modèle évoqué au paragraphe 4.2.

4 Appropriation de l'outil et apports

4.1 Difficultés de démarrage

La démarche autour de la CMDB a réellement démarré début 2013 avec une première maquette d'*iTop*. Pourtant, un réel projet n'a pu voir le jour que mi-2014. On peut s'interroger sur ce décalage. Si la CMDB était le Graal comment se fait-t-il que son implémentation ait mis autant de temps à prendre racine ? On pourrait avec facilité évoquer une charge de travail conséquente qui ne laissait que peu de place aux expérimentations mais on peut également aller plus loin dans l'analyse. Dans un premier temps, une CMDB même adossée à un ronflant volet de gestion de services (ITSM - IT Service Management) semble simplement être un outil de plus, encore un outil, qui n'apporte pas de réelle plus-value :

- des outils d'inventaires au niveau du parc ou des éléments d'infrastructure sont déjà en place ;
- les applications et les services ont probablement déjà été cartographiés par des architectes du système d'information ou des qualiticiens ;
- la présence très probable d'un outil de gestion de ticket interne qui donne satisfaction dans les grandes lignes semblent rendre caduc le volet ITSM d'*iTop*.

Considérant le coût de mise en place d'une maquette avec des données réelles, la mise en place de la CMDB ressemble à un luxe que l'on ne peut pas se permettre.

Le facteur qui a permis à la démarche d'avancer à l'Université de Strasbourg a été la volonté du responsable du support informatique de structurer et de lier de nombreuses données concernant le périmètre d'intervention de la DI. Ce périmètre d'une complexité extrême qui était préalablement décrit dans un tableur impossible à maintenir a servi de catalyseur pour le projet de mise en place de la CMDB dans la mesure où il fallait lier les services, les clients, les bâtiments et les équipements. D'une manière ou d'une autre, la présence d'un besoin non satisfait sert de mouche du coche pour ce genre de démarche. Par ailleurs, le fait de se focaliser sur ce besoin particulier peut permettre d'éviter de se disperser, ce qui est important dans le cas de la CMDB. Le projet mené à Strasbourg est parfois tombé dans le piège de la multiplicité des cibles visées. Cette dispersion provoque un effet secondaire à éviter dans le sens où cela peut allonger les délais de réalisation. La présence de mécanismes de synchronisation de données évoqués dans le chapitre 2 a également facilité la réalisation d'une maquette convaincante, c'est-à-dire présentant des données réelles.

4.2 Appropriation de l'outil

Une CMDB propose une agrégation de données existantes dans le système d'information, mais permet aussi la saisie d'éléments manquants. Cette situation provoque 2 types de problèmes différents :

- dans le cas d'éléments existants, il s'agit de les récupérer dans des bases sources diverses. Cette manœuvre peut engendrer des effets de bord : certaines données qui étaient présentes dans ces bases sources peuvent être complétées par des données saisies manuellement. Il peut s'avérer pertinent de décaler cette saisie manuelle pour qu'elle se fasse désormais dans la CMDB, mais cela implique de changer des pratiques. Dans ce cas précis, il faut mettre en avant un gain, faute de quoi ce changement n'aura pas lieu. Il faut alors trouver des réponses aux questions suivantes : que peut apporter - directement - la CMDB à ceux qui doivent alors changer leurs pratiques ? Quels gains pour tous les collègues ? La réponse à ces questions ne peut se trouver qu'en travaillant directement avec les

équipes concernées. Bien souvent, les informations étaient présentes dans d'autres outils qui ont pu être partiellement tordus pour répondre à des problématiques qui ne sont pas initialement les leurs. Ainsi le catalogue des services était-t-il modélisé dans *GLPI* qui n'est pas fait pour ça. Recentrer un outil sur ses fonctions de base facilite sa maintenance. Dans le cas d'informations qui étaient présentes dans des fichiers isolés, leur déversement dans la CMDB permet un meilleur partage et une mise à jour plus facile (automatisée) des informations. C'est le cas par exemple du fichier escalade géré par le support de la Direction Informatique qui regroupait des centaines de structures, des dizaines de correspondants, plus d'une centaine de bâtiments et de nombreux services ;

– dans le cas d'éléments nouveaux, la problématique de la conduite du changement est moindre, mais encore une fois, la définition du méta-modèle de données ne peut se faire qu'avec l'implication d'acteurs représentatifs.

De manière générale, la problématique est bien celle de la valeur ajoutée du produit pour tous. Il n'est pas forcément évident de la démontrer. Dans notre cas, des ateliers ouverts de présentation ou des sessions dédiées à certaines problématiques ont été mis en place. La difficulté est ensuite d'être capable de répondre rapidement aux attentes soulevées par ces ateliers. Dans notre cas le bât a pu blesser à ce niveau.

4.3 Gouvernance du modèle de données

Il faut s'entendre sur l'évolution du modèle de données. Dans notre contexte, le principe de gestion suivant a été défini :

– le méta-modèle de données appartient à tous. Tous peuvent proposer des évolutions de ce modèle. La version de référence est représentée par *iTop* sous la forme de fichiers *XML* qui sont stockés sur un système de gestion de versions (*git*). La création d'instances de développement d'*iTop* a été automatisée afin de faciliter les tests ;

– une proposition d'évolution du modèle de données se fait via une demande de modification du schéma. Cette demande est examinée par un collège de membres représentatifs et en cas d'absence d'objection, elle est considérée comme validée. Les architectes du Système d'Information ont bien sûr vocation à se pencher sur ces demandes ;

– la modification est déployée en production.

4.4 Processus internes

iTop intègre en interne des processus prédéfinis comme le processus de gestion de tickets et le processus de gestion des demandes de changement. Cette fonctionnalité supplémentaire peut éventuellement présenter un intérêt. Dans le cas de la Direction Informatique, le module de gestion de tickets n'a pas été activé car un outil qui donne satisfaction est déjà en place. En revanche, concernant le processus de gestion des changements, *iTop* propose un module. Il est possible de l'activer en version "lourde" ou "légère". Le workflow alors mis en place dépend du choix de l'administrateur. Ce processus présente un intérêt potentiel dans la mesure où la gestion des changements peut dès lors s'appuyer sur des informations de la CMDB (configuration). À l'Université de Strasbourg, un processus de gestion des changements est en place depuis 2012. On aurait pu penser que le module d'*iTop* aurait été une opportunité pour ce processus. Dans les faits, cela n'a pas été le cas. *iTop* propose deux workflows différents, mais la rigidité de ces derniers ne facilite pas leur utilisation réelle, surtout quand des choix ont déjà été faits. Dans notre cas, les modules proposés n'ont donc pas été utilisés, leur valeur ajoutée s'avérant négative.

De plus, *iTop* présente un défaut particulier dans ce domaine : un plugin activé à un instant donné ne peut plus être désactivé par la suite. Il devient alors délicat de faire des tests dans la mesure où ces derniers ne sont pas totalement réversibles. Nous avons contourné ce problème en mettant en place l'architecture suivante :

– une instance de test peuplée avec des données issues du système d'information
– une myriade d'instances de développement créées via des scripts qui copient l'instance de test principale
– une instance de production

4.5 Bonus

Nous allons évoquer dans ce chapitre deux bénéfices induits par la démarche et qui méritent d'être soulignés.

Le premier a déjà été évoqué, il s'agit du décloisonnement de l'information et du lien entre des informations qui étaient isolées. Des fichiers bureautiques "confidentiels" ou des bases de données utilisées exclusivement par une équipe se retrouvent exposés et tous peuvent les consulter. Même si les corrélations que l'on souhaite alors naturellement réaliser nécessitent un travail sur la qualité et la cohérence des données, ce point est en définitive bénéfique.

Le deuxième point n'a pas encore été souligné et il mérite de l'être : *iTop* présente une fonctionnalité intéressante d'*analyse d'impact*. Pour un élément de configuration (CI) donné on peut évaluer (en fonction de son type) :

– de quoi il dépend ;
– ce qu'il impacte.

On peut ainsi essayer de déterminer l'impact d'une maintenance ou d'une panne ce qui donnera la réponse à la question magique "que se passe-t-il si je casse ceci ?". *iTop* ne donne pas d'attribut aux liens de dépendances, on ne peut donc pas modéliser finement une architecture impliquant de la redondance. Cela veut dire que l'analyse d'impact ne déterminera pas automatiquement l'effet d'une indisponibilité d'un CI. L'analyse d'un ingénieur maîtrisant l'architecture reste nécessaire, mais c'est probablement mieux ainsi. L'analyse d'impact permet d'attirer l'attention sur un impact potentiel, que ce dernier soit maîtrisé par une architecture redondante ou pas.

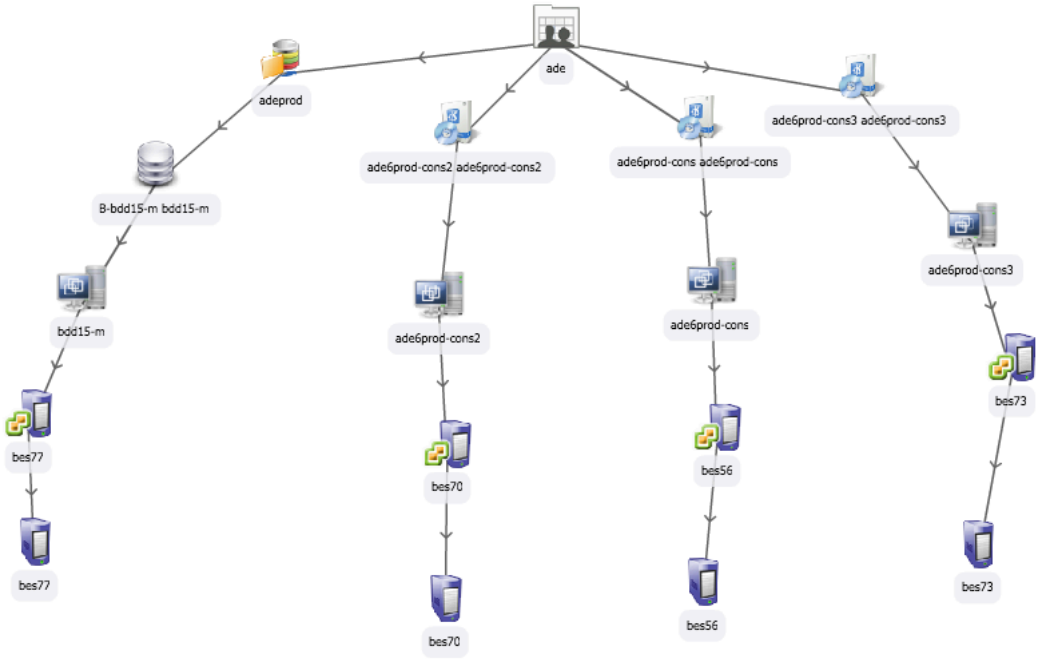

Figure 6 - Arbre de dépendance

Une telle démarche nécessite de modéliser et synchroniser l'ensemble des CI nécessaires au fonctionnement d'un service. Toutefois, elle permet de gagner en maîtrise du système d'information et de mieux cibler la communication vers les utilisateurs en cas de panne.

5 Conclusion

On le voit dans cet article, la mise en place d'une CMDB n'est pas un Graal inaccessible. Toutefois, le projet de mise en place d'un tel outil n'est pas trivial et il a des impacts multiples et nécessite une forte intégration au système d'information. Justement, quand la route est longue, et dans la mesure où la CMDB présente de nombreuses facettes, il convient de se focaliser sur des objectifs réalistes et d'avancer par étapes, de découper le SI en morceaux afin de présenter des résultats intermédiaires qui permettent de démontrer les apports de la démarche.