# Retour d'expérience

# La personnalisation du logiciel libre de gestion d'incidents « Request Tracker »

*Par Christophe NOWICKI*
<*cnowicki@easter-eggs.com*>

Janvier 2006

# Table des matières

1 - Introduction....3
1.1 - Introduction à la gestion d'incident....3
1.2 - Solution libre « Request Tracker »....4
1.2.1 - Fonctionnalités....4
1.2.2 - Points fort : la personnalisation....4
1.2.3 - Points faible : la complexité....5
2 - Analyse des besoins et des solutions proposées....6
2.1 - Gestion Qualité Industrielle....6
2.2 - Pertinence de l'utilisation d'un système de gestion d'incident....6
2.3 - Personnalisation de la solution....7
2.3.1 - Adaptation du processus....7
2.3.2 - Exploitation des informations....8
3 - Les problématiques techniques rencontrées et leurs solutions....9
3.1 - Description de l'architecture du logiciel....9
3.2 - Problématiques techniques et les solutions mise en oeuvre....10
3.2.1 - Le piège des « Custom Fields »....10
3.2.2 - Complexité du moteur de recherche....10
3.2.3 - Vision métier de la base de données....11
3.3 - Quelques idées d'amélioration et projets intéressants....12
3.4 - Améliorations....12
3.4.1 - Performances....12
Amélioration de l'API de récupération des objets dans la base de données....12
Choix d'un autre algorithme pour prendre en charge les champs personnalisés....12
3.4.2 - Ergonomie....12
Utilisation de requête HTTP asynchrones (AJAX)....12
Création d'un client riche en XUL....12
3.5 - Idées de projets intéressants....13
3.5.1 - Fédération d'identité....13
3.5.2 - Intégration avec un PABX....13
4 - Conclusion....14

# 1 - Introduction

L'objectif de ce document est de partager l'expérience que nous avons acquise lors de la transformation d'un logiciel de gestion d'incident (Request Tracker) en un système de gestion de qualité industrielle.

Il s'adresse aux responsables et aux techniciens qui souhaitent mettre en place une solution de gestion d'incidents fondée sur le logiciel libre RT.

## *1.1 - Introduction à la gestion d'incidents*

La gestion d'incidents consiste à :

- alerter et déclarer ;
- valider et répondre ;
- suivre et chiffrer les incidents.

De nombreux professionnels ont besoin de ce type de fonctionnalités, notamment :

- dans les centres de support clients ;
- dans la gestion de parc informatique « Help desk » ;
- dans l'industrie.

Chaque profession dispose de son propre vocabulaire mais le processus de gestion est plus ou moins le même :

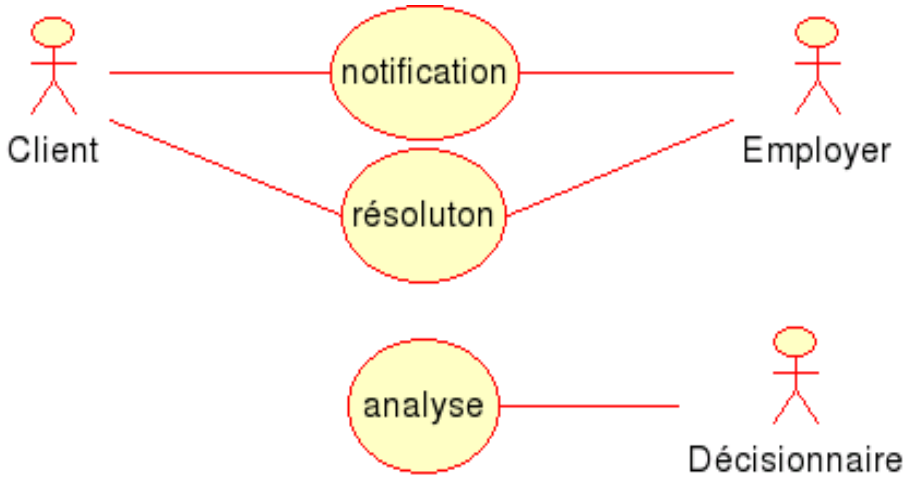



*Diagramme de cas d'utilisation*

## *1.2 - Solution libre « Request Tracker »*

*Logo de Best Pratical*

Request Tracker est un logiciel libre édité par la société « Best Pratical Solutions »[1] sous licence GPL[2].

Il est composé de plusieurs éléments :

- un moteur de gestion de « tickets »
- une base de données relationnelle
- plusieurs interfaces : web, ligne de commande et messagerie.

### 1.2.1 - Fonctionnalités

Les principales fonctionnalités du logiciel sont les suivantes :

- déclarer, assigner, organiser, rechercher et répondre à un incident ;
- la déclaration peut se faire via l'interface web ou bien via l'envoi d'un mail ;
- historique complet des actions et de toutes les modifications entreprises ;
- interface multi-utilisateurs et traduction en de nombreuses langues ;
- support de nombreux types de bases de données relationnelles (MySQL, Oracle, PostreSQL, etc...) ;
- décompte du temps consacré ;
- moteur de recherche / extraction des informations.

### 1.2.2 - Point fort : la personnalisation

La plus grande force du logiciel, est la facilité de personnalisation :

- champs personnalisables (« Custom Fields »), il s'agit d'attribuer des champs de différents types à un événement ;
- scripts (« Scripts »), il s'agit d'un système de macros déclenchées en fonction d'un événement. Par exemple, il est possible d'envoyer un mail à un responsable lorsqu'un certain type d'incident est signalé ;
- une multitude d'options de configuration ;
- une architecture modulaire.

1 http://www.bestpractical.com
2 http://www.gnu.org/copyleft/gpl.html

### 1.2.3 - Point faible : la complexité

- l'installation n'est pas des plus aisée et demande des compétences non négligeables en informatique, il faut donc compter une bonne journée ;
- l'application est gourmande en ressources, il est nécessaire de la déployer sur une machine relativement puissante avec des ressources disques et mémoire conséquentes et même sur une bonne machine, il faut prendre le temps de « Tuner[3] » le système pour obtenir de bonnes performances ;
- la charte graphique par défaut n'est pas très jolie, mais c'est un point assez subjectif ;
- l'ergonomie de l'interface graphique n'est pas ce qu'ont fait de mieux en la matière. Certains écrans sont fouillis et très complexes à utiliser. De plus, les contraintes liées aux formulaires web n'arrangent pas les choses.

Toutefois, RT est le logiciel libre de référence en matière de gestion d'incidents. Il est utilisé par de nombreuses entreprises[4] ou organisations parmi lesquelles : la NASA, Merrill Lynch, Freshmeat, Free Telecom, etc...

3 http://wiki.bestpractical.com/index.cgi?PerformanceTuning
4 http://www.bestpractical.com/rt/praise.html

# 2 - Analyse des besoins et des solutions proposées

## *2.1 - Gestion Qualité Industrielle*

Les missions du service Qualité du Client sont multiples :

- Alerter/informer d'une non-conformité ;
- Mettre en place des actions correctives afin de remédier à ces non-conformités ;
- Mettre en place des actions préventives ;
- Chiffrer les coûts des non-conformités et des actions de correction et prévention qu'elles engendrent.

Cette gestion nécessite de désigner des responsables d'action et de fixer des délais de réalisation. L'ensemble de ces éléments doivent être organisés et hiérarchisés pour mieux en assurer le traitement et le suivi.

## *2.2 - Pertinence de l'utilisation d'un système de gestion d'incident*

Partant du principe qu'une gestion de non-conformité et une gestion d'incidents répondent aux mêmes principes de traitement, l'adaptation de RT à une problématique Qualité devient alors une question de sémantique.

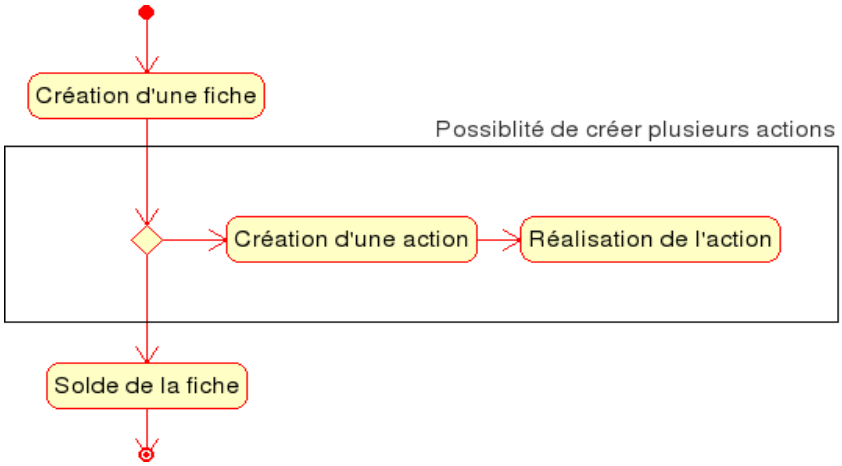
On ne parle plus d'incident, mais de non-conformité, on ne dit plus ticket, mais fiche qualité, un incident n'est plus résolu, mais une fiche est soldée, etc.

Néanmoins, il reste pertinent (et nécessaire) de se pencher sur les spécificités du Client en exploitant le fait que RT est très personnalisable.

En effet, l'objectif est, dans le mesure du possible, d'adapter l'outil aux pratiques du Client et non de le contraindre à une logique induite par un fonctionnement trop rigide du logiciel.

## *2.3 - Personnalisation de la solution*

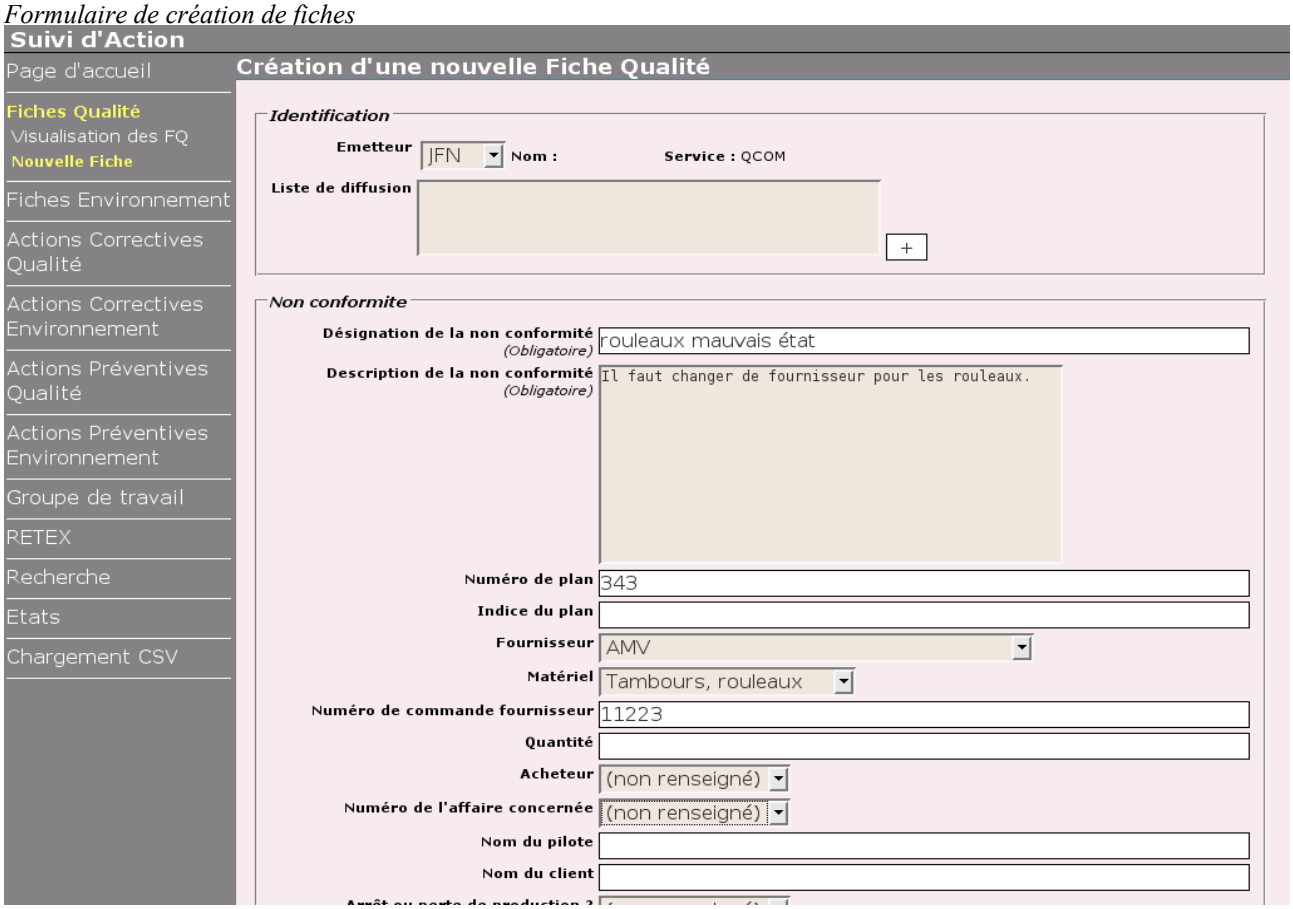
### 2.3.1 - Adaptation du processus




*Processus de gestion qualité*

Pour chaque type d'incident, une fiche est créée. Celle-ci se présente sous forme d'un formulaire permettant d'identifier l'incident et d'informer les responsables par le biais d'une liste de diffusion :

*Formulaire de création de fiches*



Chaque fiche donne lieu à la mise en place de plusieurs actions avec pour chacune un responsable.

Une fois l'ensemble des actions qui composent une fiche réalisée, celle-ci est soldée.

Au cours de ce processus, les utilisateurs ont la possibilité :

- d'échanger des fichiers ;
- de communiquer par écrit.

Le service Qualité n'a plus qu'à valider l'ensemble des actions et à exploiter les informations fournies.

### 2.3.2 - Exploitation des informations

Pour exploiter l'ensemble des informations fournis, les utilisateurs disposent d'un moteur de recherche simplifié et d'un système d'extraction de statistiques : le moteur d'état.

Celui-ci permet principalement de calculer les coûts des incidents et de les afficher sous forme de tableaux :

Imprimer cette page

| | Service | Nombre de FQ | % par rapport au totale | Coût en € | % par rapport au coût |
|---|---|---|---|---|---|
| 1 | ACH | 8 | 1.3 % | 42060 € | 20.4 % |
| 2 | AT | 154 | 24.7 % | 16444.1 € | 8.0 % |
| 3 | BEA | 35 | 5.6 % | 17182 € | 8.3 % |
| 4 | BEM | 42 | 6.7 % | 4360 € | 2.1 % |
| 5 | DAF | 3 | 0.5 % | 0 € | 0.0 % |
| 6 | DCI | 4 | 0.6 % | 2040 € | 1.0 % |
| 7 | DG | 5 | 0.8 % | 0 € | 0.0 % |
| 8 | ETF | 8 | 1.3 % | 14260 € | 6.9 % |
| 9 | EXM | 12 | 1.9 % | 1362 € | 0.7 % |
| 10 | FAA | 6 | 1.0 % | 300 € | 0.1 % |
| 11 | MMS | 33 | 5.3 % | 23373 € | 11.3 % |
| 12 | PILA | 13 | 2.1 % | 13889 € | 6.7 % |
| 13 | PILI | 21 | 3.4 % | 21679.99 € | 10.5 % |
| 14 | QCOM | 229 | 36.8 % | 26270.21 € | 12.7 % |
| 15 | SER | 45 | 7.2 % | 22990.8 € | 11.1 % |
| 16 | TCA | 5 | 0.8 % | 0 € | 0.0 % |
| 17 | **TOTAL période** | 623 | | 206211.1 € | |

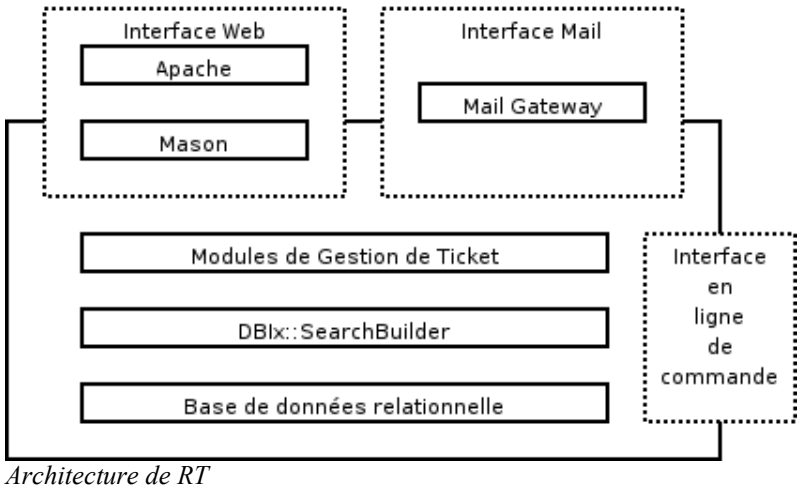
Done

*Illustration 1: Exemple d'états*

# 3 - Les problématiques techniques rencontrées et leurs solutions

## *3.1 - Description de l'architecture du logiciel*

Voici le schéma de l'architecture générale du logiciel. L'ensemble est écrit à l'aide du langage de programmation Perl.




*Architecture de RT*

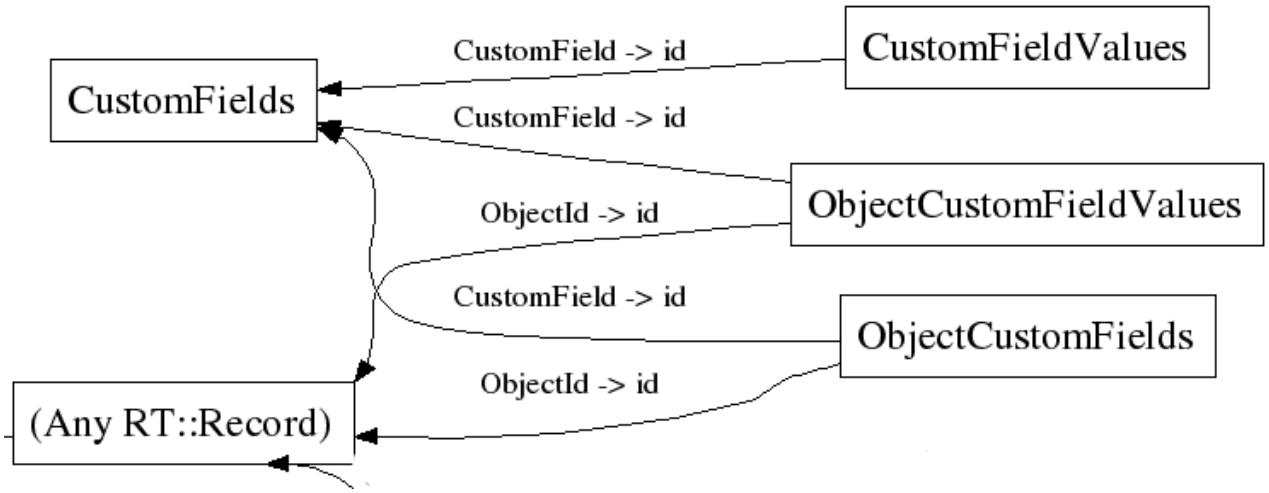
- Base de données relationnelle, le logiciel dispose d'une couche d'abstraction par rapport à la base de données grâce aux modules DBI[5] ;
- Les requêtes effectuées sur la base sont construites par le biais du module DBIx::SearchBuilder[6] ;
- Le coeur du logiciel est composé d'un ensemble de modules / classes Perl : Ticket, User, Group etc.
- Au dessus de ces modules sont greffées trois interfaces :
  - en ligne de commande ;
  - via la messagerie, il s'agit d'un script qui fonctionne comme le programme procmail[7] ;
  - web, celle-ci est écrite à l'aide du système de modèle Mason[8]. Il s'agit d'un framework permettant la réalisation de site web en Perl. Cette interface fonctionne de plusieurs manières (CGI, FastCGI[9], mod_perl[10]) à l'intérieur d'un serveur HTTP comme Apache.

5 http://dbi.perl.org/
6 http://search.cpan.org/~jesse/DBIx-SearchBuilder-1.36/SearchBuilder.pm
7 http://www.procmail.org/
8 http://www.masonhq.com/
9 http://www.fastcgi.com/
10 http://perl.apache.org/

## *3.2 - Problématiques techniques et solutions de mise en oeuvre*

### 3.2.1 - Le piège des « Custom Fields »

Voici la structure utilisée pour prendre en charge les champs libres dans RT :




*Structure utilisée pour prendre en compte les "customfields" dans RT*

Cette structure est inspirée par les algorithmes de persistance objet[11], en effet les « custom fields » sont vus comme des attributs pour les objets (Ticket, Queues, etc.).

Le principal problème de l'algorithme choisi est qu'il ne monte pas en charge, en effet il faut faire de nombreuses requêtes SQL pour obtenir la valeur d'un champ libre.

D'autres algorithmes, plus efficaces, mais plus complexes sont décrits dans le livre de référence[12].

Par conséquent, il ne faut pas abuser de cette fonctionnalité et limiter le nombre de champs personnalisés, dix nous semble être le maximum.

Si vous voulez étendre les attributs des différents objet, il est préférable de modifier le schéma de la base de données.

### 3.2.2 - Complexité du moteur de recherche

Le moteur de recherche disponible par défaut dans le logiciel est extrêmement puissant. Il permet la création de requêtes « Ticket SQL », forme de SQL simplifiée permettant de rechercher dans la base de données. Le problème est que ce moteur de recherche est inutilisable, car trop complexe.

Un utilisateur, ne saura jamais s'en servir, même après une longue formation.

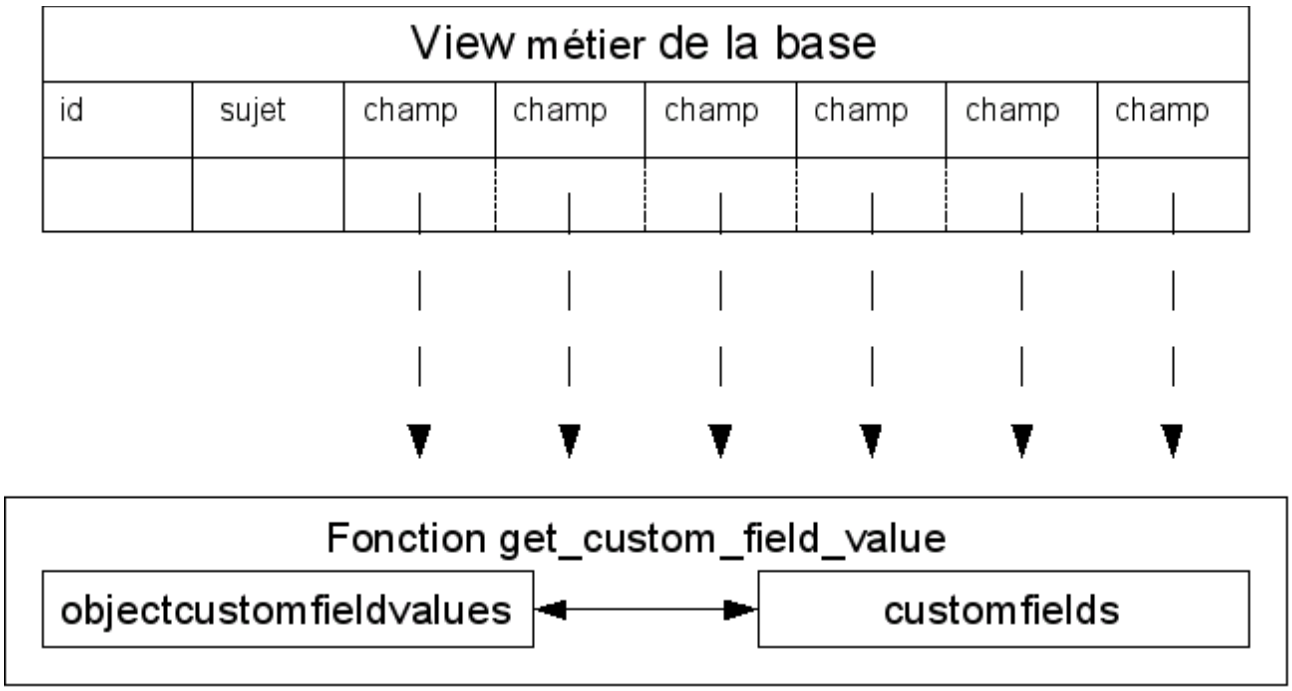
Nous avons donc récrit, un moteur de recherche simplifié. Avec un accès rapide aux recherches les plus fréquentes et un choix limité aux critères les plus importants.

11 http://en.wikipedia.org/wiki/Object-relational_mapping
12 http://www.ambysoft.com/books/agileDatabaseTechniques.html

### 3.2.3 - Vision métier de la base de données

Le client avait besoin de voir et d'extraire les informations de la base de données. Dans un premier temps nous lui avons donné accès à la base de données du logiciel, mais cela est relativement complexe, notamment pour récupérer les valeurs contenues dans les champs personnalisés.

Pour contourner le problème nous avons mis en place un système de « View ». Celui-ci fonctionne de la manière suivante :




*Système de "Views"*

La base de données récupère les valeurs des champs personnalisés à l'aide d'une fonction PL/SQL, ce qui permet à l'utilisateur d'effectuer des interrogations sur les données. Les modifications sont prise en compte par une fonction.

## *3.3 - Quelques idées d'amélioration et projets intéressants*

## *3.4 - Améliorations*

### 3.4.1 - Performances

#### *Amélioration de l'API de récupération des objets dans la base de données*

Le « framework » RT, ne propose qu'une seule fonction pour récupérer un objet dans la base de données : *Load*.

Il est dommage que cette fonction ne permette pas de préciser quels sont les attributs / éléments dont nous avons besoin.

Par conséquent, il arrive souvent de charger un objet complexe, comme un ticket, pour seulement en afficher un ou deux attributs.

#### *Choix d'un autre algorithme pour prendre en charge les champs personnalisés*

Une autre algorithme que celui qui est utilisé actuellement dans l'application permettrait de stocker bien plus d'informations dans les champs personnalisés.

### 3.4.2 - Ergonomie

L'interface de RT est complexe et donc difficile à prendre en mains, voici quelques pistes pour rendre son interface web plus agréable :

#### *Utilisation de requête HTTP asynchrones (AJAX)*

Cette méthode est actuellement à la mode, car elle repousse les limites des possibilités offertes par les technologies du Web. Lors de la modification de RT, nous avons ajoutés de nombreuses fonctionnalités dynamiques telle que la complétion automatique, la validation lors de la saisie des formulaires, etc...

Nous pensons que l'architecture globale de l'application est bien pensée et par conséquent l'utilisation de cette méthode peut apporter beaucoup dans l'ergonomie de l'application.

Elle gomme aussi l'impression de lourdeur qui est liée à l'utilisation de formulaire web classique.

#### *Création d'un client riche en XUL*

Les fonctionnalités offertes par l'interface du logiciel sont nombreuses et les limites des technologies web se font vite sentir. L'adaptation de la technologie XUL[13] permettrait de fournir une interface graphique riche disposant des fonctionnalités classiques telles que : les menus contextuels, le glisser/déposer et des contrôleurs complexes.

Le projet est moins ambitieux qu'il n'y paraît, il faut principalement décrire l'interface web

13 http://www.mozilla.org/projects/xul/

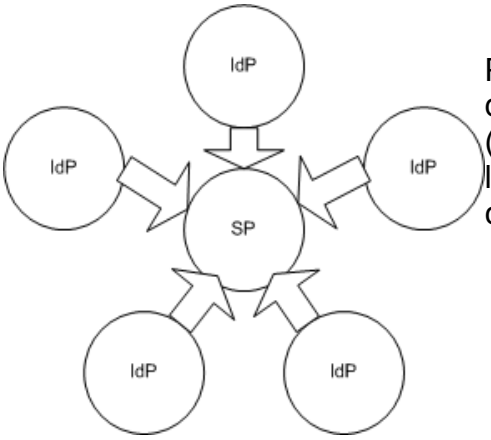
en XUL et fournir les données (objets, et recherches) au format RDF[14].

## *3.5 - Idées de projets intéressants*

### 3.5.1 - Fédération d'identité

Voici le scénario qui pourrait être réalisé grâce au support des spécifications « Liberty Alliance » [15] dans RT.

« Vous êtes le fournisseur d'une entreprise, vous vous identifiez auprès de votre système d'information, puis vous vous connectez sur l'interface de gestion d'incidents de votre client pour signaler un problème. Sans aucune intervention de votre part l'ensemble des informations vous concernant sont remplies dans l'interface... ».



*Topologie de type "Service Provider HUB"*

Pour réaliser ce scénario, il faudrait ajouter le support de la bibliothèque Lasso[16] pour la fédération d'identité (ID-FF[17]) et l'échange d'attributs (ID-WSF[18]) dans RT et le faire communiquer avec un fournisseur identité, comme Authentic[19].

### 3.5.2 - Intégration avec un PABX

Le téléphone demeure l'outil le plus exploité pour effectuer une demande. En moyenne, 46% des demandes sont réalisées par ce canal, la messagerie arrive juste derrière avec 44%. Par conséquent, si vous voulez prendre en charge le plus efficacement possible les incidents, il faut inévitablement marier ces outils : la téléphonie, la messagerie et le système de gestion d'incident.

Request Tracker, prend très bien en charge la messagerie, mais néglige le téléphone.

Le mariage entre le système de PABX Asterisk[20] et RT semble être très intéressant [21].

14 http://www.w3.org/RDF/
15 http://www.projectliberty.org/
16 http://lasso.entrouvert.org/
17 http://www.projectliberty.org/resources/specifications.php#box1
18 http://www.projectliberty.org/resources/specifications.php#box2a
19 http://authentic.labs.libre-entreprise.org/
20 http://www.asterisk.org/
21 http://www.marko.net/asterisk/archives/0210/0107.html

# 4 - Conclusion

Lors de la modification de logiciel RT, nous avons beaucoup appris.

Ce logiciel propose diverses fonctionnalités et peut répondre à de nombreux besoins en entreprise.

Si vous vous apprêtez à modifier votre installation ou étendre de RT, j'espère que cet article vous apportera des informations intéressantes.

Je vous conseille vivement de réfléchir sur la manière dont vous aller vous y prendre, car il suffit parfois de deux ou trois lignes de code pour complètement modifier le comportement du programme.