LE CENTRE
DE SERVICES PARTAGÉS
DU QUÉBEC

Communautés de pratique en logiciel libre
Modèle de fonctionnement

Centre d'expertise en logiciel libre
800, Place d'Youville,
Québec (Qc) G1R 3P4

2013-10-23
Version 1.0

Préambule

Ce présent modèle de fonctionnement est le résultat de plusieurs rencontres de travail qui regroupe plusieurs parties prenantes des différents organismes publics dont les auteurs sont :

Hakim Elhadjen hakim.elhadjen@cspq.gouv.qc.ca	Centre d'expertise en logiciel libre
Claude Durocher	Centre d'expertise en logiciel libre
Luc Lessard	Ministère de la Sécurité publique
Nicolas Gignac	Ministère de la Sécurité publique
Stéphane Cyr	Ministère de la Sécurité publique
Nicolas Brisebois-Tétreault	Ministère de la Sécurité publique
Marc-André Trottier	Ministère de la Sécurité publique
Steve Toutant	Institut de la Santé publique
François Gourdeau	Ministère de la Culture et des Communications
Daniel Morissette	MapGears

Table des matières

1 Introduction

Bien avant de focaliser sur les outils de collaboration et de partage de connaissances, il est primordial de définir un modèle de fonctionnement qui est à la fois un règlement et un guide de bonnes pratiques afin d'assurer une meilleure coordination entre les différentes parties prenantes d'une communauté.

Plan de travail :

- Identification des parties prenantes (exemples : qui définit les orientations? Qui prend les décisions? Qui peut devenir développeur ? Etc.);
- Définition du processus de fonctionnement de la communauté (exemple : comment doivent se prendre les décisions ? Comment le transfert de technologies entre les entités administratives doit se faire ? Etc.);
- Réflexion autour de la solution technologique (outils de collaboration, plusieurs formes, dont la mise en place de forums et de listes de discussion, de sites Web spécialisés) à privilégier afin de rendre cette collaboration opérationnelle.

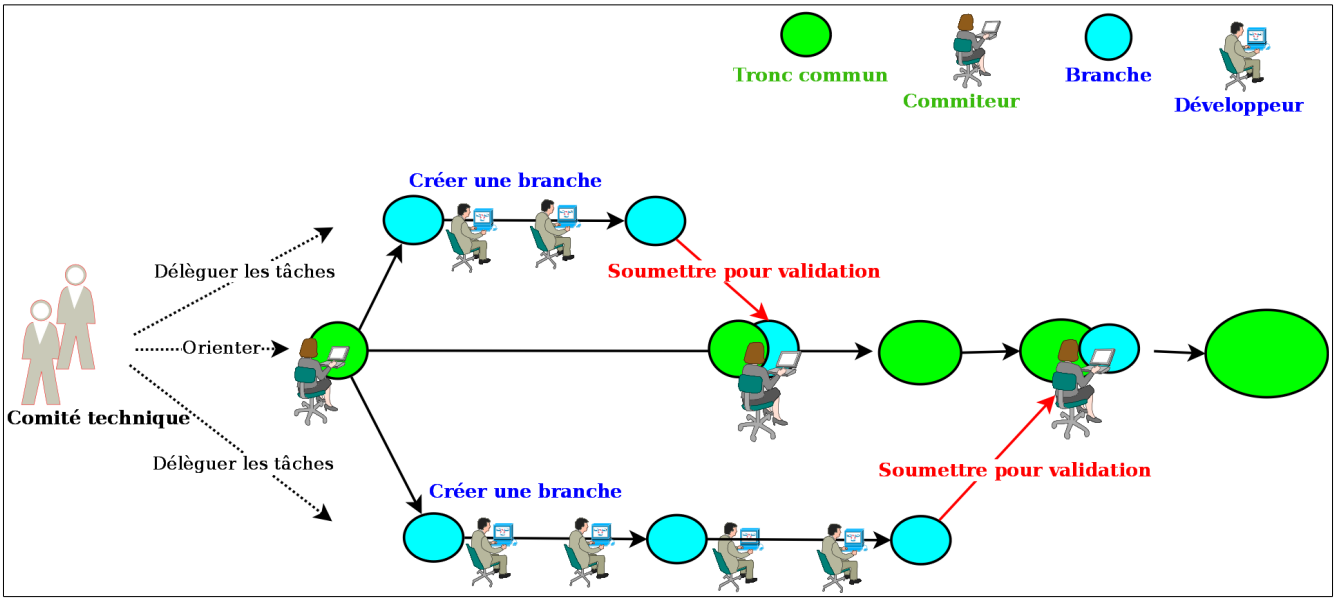
Quelques concepts techniques :

Tronc commun « master » : La version stable du logiciel, testée et validée par les commiteurs.

Commiteurs : Ce sont des programmeurs expérimentés habilités à valider le travail des développeurs.

Branche : Une copie du tronc commun que les développeurs créent localement.

Afin d'assurer une meilleure collaboration de plusieurs personnes sur un même projet tout en garantissant une meilleure stabilité du tronc commun, l'utilisation des branches est indispensable. Tous les développeurs qui souhaitent participer à l'amélioration du projet créent des branches, localement, sur lesquelles ils peuvent faire des tests. Les améliorations sont ensuite soumises aux commiteurs pour les réintégrer au tronc. Ces derniers doivent prendre une décision quant à leur réintégration.

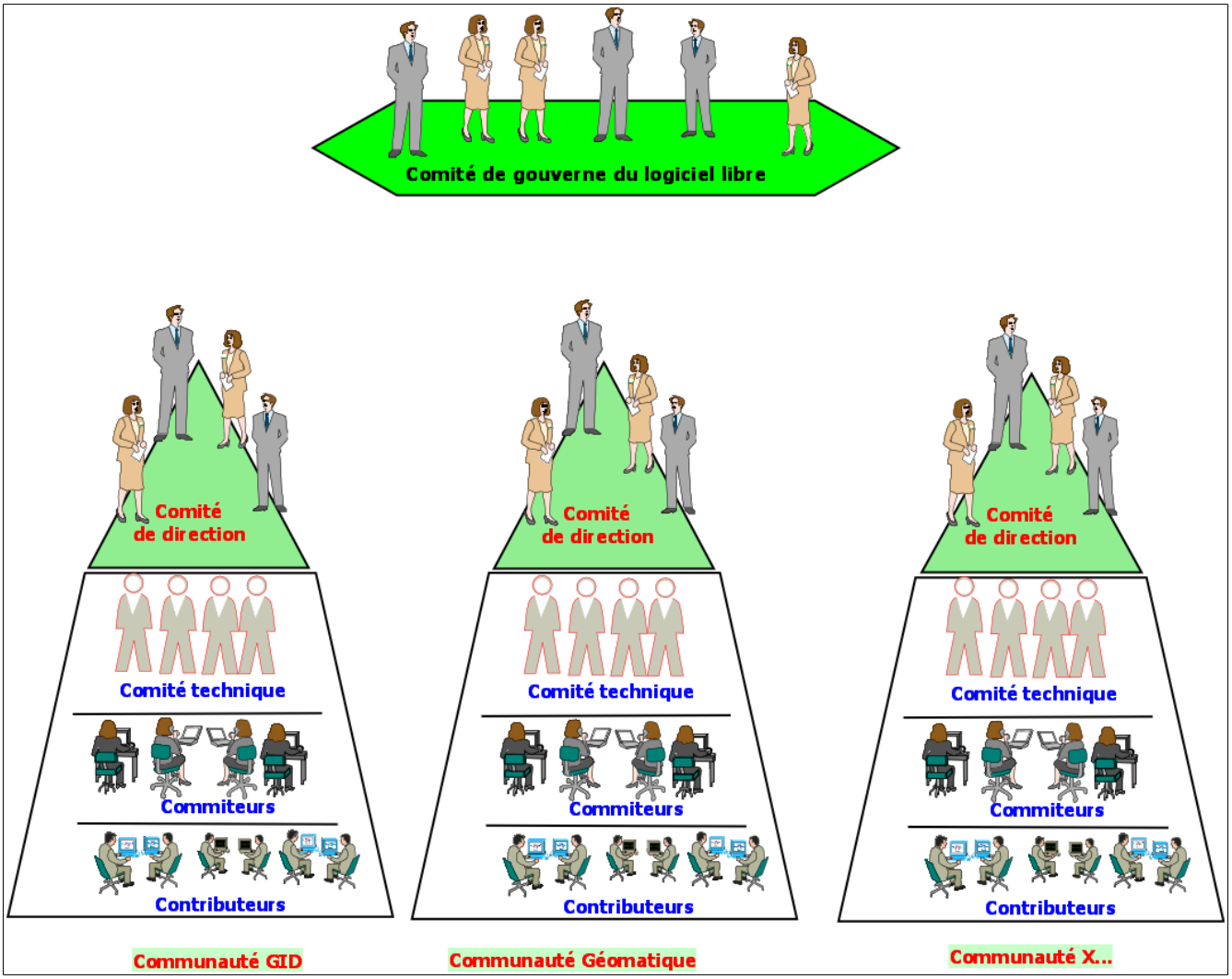


2 La licence

Le volet des licences est en cours d'étude au Secrétariat du Conseil de Trésor (SCT).

3 Identification des parties prenantes et des responsabilités de chacune

Un **comité de gouverne** pour l'ensemble des communautés gouvernementales définit les orientations stratégiques des projets. Au sommet de chaque communauté, il y a un **comité de direction** et un **comité technique** qui pilote et veille au bon fonctionnement du projet. Ce dernier choisit ensuite des **commiteurs** pouvant garantir la stabilité du logiciel. Les **contributeurs** des différents organismes publics qui partagent des besoins communs forment la base de chaque communauté.



3.1 Comité de gouverne du logiciel libre

Il s'agit du comité de gouverne du logiciel libre dont la vice-présidence aux solutions d'affaires (VPSA) est chargée d'animer.

3.1.1 Responsabilités:

- Émettre des recommandations au Centre de service partagé du Québec (CSPQ) sur les orientations, activités et projets du centre d'expertise en logiciel libre (CELL) ainsi que sur les priorités à accorder à ceux-ci en fonction des besoins gouvernementaux;
- Effectuer le suivi des huit mesures en logiciel libre ainsi que l'état d'avancement des projets phares;
- Faciliter l'utilisation du logiciel libre au gouvernement du Québec.

3.2 Comité de direction

Autorité ultime du projet.

3.2.1 Composition :

Gestionnaires responsables habilités à assigner des ressources au projet.

3.2.2 Responsabilités:

- Assigner des ressources au projet;
- Entériner les conventions et politiques de travail du projet (en concertation avec le comité technique);
- Entériner la vision générale du produit "roadmap" (en concertation avec le comité technique);
- Gérer la relation avec le comité de gouverne, les ministères et le monde extérieur;
- Définir les priorités du projet, conjointement avec le comité technique;
- Produire un rapport annuel du projet au comité de gouverne.

3.3 Comité technique

Les organismes publics qui s'implique dans un projet doivent fournir un répondant (représentant) qui veillera à la bonne compréhension de leurs besoins et des orientations. Les décisions se prennent par consensus, par un système de vote :

- Il relève de comité de direction;
- Président ("chair") élu parmi les membres du comité;
- Éviter qu'un seul organisme domine le comité;
- Viser cinq ou sept membres au démarrage pour un bon équilibre et permettre l'expansion.

3.3.1 Composition :

- Architectes intégrateurs / fonctionnels;
- Analystes fonctionnels;
- Développeurs expérimentés;
- Pilotes.

3.3.2 Responsabilités:

- Entériner les conventions et politiques de travail du projet (en concertation avec le comité de direction);
- Entériner la vision générale du produit “roadmap” (en concertation avec le comité de direction);
- Coordonner la publication de révisions régulières du logiciel (plan de livraison);
- Réviser et approuver les demandes de changement (RFC);
- Gérer les droits d'accès aux services;
- Gestion de l'infrastructure du projet (git/svn, site web, etc) (sera prise en charge par la forge gouvernementale éventuellement);
- Gérer les statuts des commiteurs;
- Définir les priorités du projet, conjointement avec le comité directeur;
- Produire un rapport annuel du projet au comité de direction.

3.3.3 Élection de nouveaux membres

- Au besoin, ou lorsqu'on bon candidat se démarque, il peut être nominé et élu par motion et vote +1 de la majorité absolue des membres existants. Démission : Un membre peut démissionner en tout temps;
- Un membre inactif pour une période prolongée (aucune participation aux votes, réunions et autres activités du comité) peut être remplacé par vote des autres membres du comité.

3.4 Commiteurs

Ce sont des programmeurs expérimentés habilités à valider le travail des contributeurs. Les commiteurs sont désignés par le comité technique. Un commiteur doit être expert et s’engage à garantir une meilleure stabilité du projet.

3.4.1 Qualifications :

- Être spécialiste du domaine;
- Avoir la confiance du comité technique et de s’engager à consacrer suffisamment du temps afin de garantir l’évolution du logiciel.

3.4.2 Privilège:

Permission de contribuer au dépôt de code source directement.

3.4.3 Responsabilités:

- S'assurer de l'intégrité des contributions (provenance, propriété intellectuelle, brevets, licence, etc.);
- Supporter ses contributions de façon raisonnable (bugs, etc.);
- Respecter les règles d'engagement;
- Signer et respecter l'entente de contribution.

3.4.4 Statut :

- Le comité technique du projet vote et approuve l'élection de nouveaux commiteurs. Il gère et active aussi les droits dans le dépôt de code source (git, svn , etc.);

- Un commiteur inactif ou qui ne respecte pas les règles d'engagement peut se faire révoquer son titre et son droit de commiteur.

3.4.5 Comment y accéder?

- Commencer par contribuer régulièrement des “patchs” via le “bug tracker”;
- Un ou des commiteurs vont réviser et endosser ces “patchs” pour inclusion officielle dans le logiciel. Cela permet de gagner la confiance des autres commiteurs;
- Confirmer le désir de devenir commiteur et de respecter les règles d'engagement;
- À ce moment, une motion sera faite au comité de direction du projet qui passera au vote afin d’attribuer le titre de commiteur.

3.4.6 Comment gagner la confiance des autres commiteurs?

- Bonne compréhension de l'architecture, des outils et méthodes de fonctionnement du projet;
- Code de qualité;
- Aptitudes de communication;
- Intention de rester actif à moyen long terme;
- Pourrait passer à travers une formation ou “examen par les pairs” pour devenir “committeur” plus rapidement au besoin.

3.4.7 Règles d'engagement du commiteur

3.4.7.1 Entente de contributions

- Vise à protéger l'intégrité du code source du logiciel contre les contributions illégitimes, accidentelles ou non, et leurs conséquences;
- Engagement légal du commiteur envers le projet confirmant qu'il a le droit de contribuer sa propriété intellectuelle (PI) au projet;
- L’entente doit être signée par tous les commiteurs ou contributeurs réguliers;
- Peut ou non inclure un transfert de droit;
- Selon les cas, une entente individuelle et corporative peut être requise (si l'employeur possède les droits sur le travail de l'employé).

3.4.7.2 Convention de codage et règle de conduite

- Visent l'uniformité du code et des méthodes de travail. Par exemple:
  - Règles d'écriture de code source;
  - Règles d'architecture, du modèle de données.
- À respecter par le commiteur, prérequis à toute contribution.

3.5 Contributeurs

Utiliser “contributeur” pour être plus inclusif.
Les contributeurs incluent:

- Développeurs / programmeurs;
- Architectes;
- Rédacteurs documentation;

- Testeurs / validateurs;
- Pilotes / utilisateurs d'application;
- Tout autre type de contribution.

4 Processus de gestion d'un projet logiciel libre ouvert

4.1 Mécanisme de décisions

- En priorité : Discussion ouverte et recherche de consensus;
- Utiliser la liste publique en tout temps (sauf dans les cas extrêmes exigeant la confidentialité);
- Dans 99 % des cas, le vote sert seulement à confirmer le consensus et officialiser la décision;
- Mécanisme de vote : +1, 0, -1.

4.1.1 Mécanisme de vote

- Chaque membre du comité a droit à un vote. Les motions et votes se font en public sur la liste de discussion;
- Le vote est habituellement ouvert pour une période relativement courte (ex. : trois jours ouvrables) sauf s'il y a exception;
- Vote +1, 0, -1:
  - +1: Je suis d'accord et m'engage à supporter cette décision et collaborer à sa réalisation;
  - 0: Abstention, sans effet (aussi sans effet: -0 légèrement en désaccord et +0 légèrement d'accord, mais avec des doutes);
  - -1: Objection, veto, doit fournir une avenue de solution alternative.
- Une proposition est acceptée si elle reçoit au moins +2 (incluant l'auteur) et aucun veto (-1);
- Si une proposition reçoit un veto (-1) et qu'il est impossible de satisfaire toutes les parties après révision et discussion :
  - La proposition peut être soumise pour un second vote ultime;
  - Dans ce cas un vote +1 de la majorité absolue de tous les membres du comité est requis pour que la proposition soit acceptée (et non pas seulement la majorité des votes soumis).
- Le résultat du vote est compilé et publié par son auteur. Il est archivé sur la liste de discussion et dans les documents associés, s'il y a lieu (RFC).

4.2 Gestion des changements (RFC)

- Processus de demande de changement (RFC) :
  - Discussions préliminaires sur la liste -dev;
  - Production d'un document RFC dans le dépôt de RFC du projet (qui, quand, quoi, pourquoi, comment, incompatibilités, etc.);
  - Discussion du RFC;
  - Vote du comité technique;
  - Auteur de la RFC reflète le résultat du vote dans le document;
  - Implémentation, documentation, etc.
- Liste des RFCs disponibles sur le site Web (ou autre dépôt);
- Le projet devrait se définir un gabarit de RFC.

4.3 Gestion des révisions

- Dépôt d'un plan de livraison;
- Rôle de responsable de livraison (doit être committeur);
- Cycle de révisions :
  - Cycle de plus au moins six mois entre les révisions, ou basé sur feuille de route (“roadmap”);
  - Dév. -> Plan livraison -> Gel de fonctionnalités -> bêtas -> RC -> livraison.

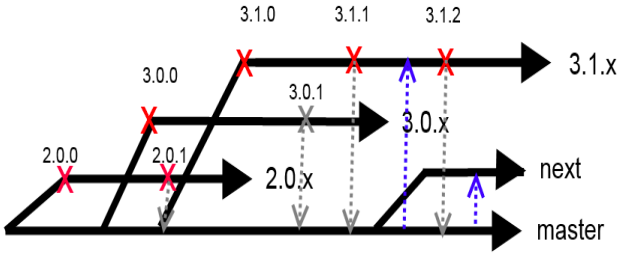
4.3.1 Numérotage des révisions

Exemple suggéré :

- Version = x.y.z (majeur.mineur.patch);
- Mineur pair = stable (ex: 5.4, 5.6);
- Mineur impair = développement (ex: 5.5);
- Patch = résolutions bogues = 5.4.1, 5.4.2, etc.

4.3.2 Branchement du code source

Exemple suggéré :



5 Infrastructure technique – forge gouvernementale

Une forge désigne un système de gestion de développement collaboratif de logiciel, c'est-à-dire un ensemble d'outils dédié au développement d'un ou de plusieurs logiciels.
Les outils nécessaire pour constituer une forge gouvernementale sont:

- Un système de gestion des versions;
- Un logiciel de gestion continue;
- Un système de gestion des projets;
- Un gestionnaire de dépôt de librairie;
- Un système d'authentification;
- Un gestionnaire de listes de diffusion.

5.1 Système de gestion des versions[1]

C'est un système qui permet d'archiver et conserver les différentes étapes de développement d'un projet. Un tel système doit permettre la création des branches afin de travailler en parallèle sur plusieurs fonctionnalités du code. L'objectif est de garder une branche principale dite tronc commun « master » stable et de créer des branches (des copies de la principale) sur lesquelles on travaille et on teste localement avant de les réintégrer au tronc commun par l'intermédiaire des commiteurs.
Cette pratique nécessite un outil de gestion des versions décentralisées[2] et l'outil retenu est **Git**[3]**.**

5.1.1 Visualisation des dépôts

Pour pouvoir visualiser facilement les dépôts Git et organiser les projets en permettant de rapporter toute erreur ainsi que de soumettre des «patchs» correctifs de manière automatisée, **GitLab**[4] est retenu pour cette fonctionnalité.

5.2 Logiciel d'intégration continue[5]

C'est un outil qui permet de vérifier automatiquement si le code compile[6] bien et s'il passe une série de tests avec succès.

Les logiciels libres qui pourraient être utilisés sont :

- **GitLab CI;**
- Jenkins ou Hudson, serveurs d'intégration continue pour Java;
- Tinderbox, serveur d'intégration continue de la Mozilla Foundation;
- Apache Continuum, serveur de l'Apache Software Foundation.

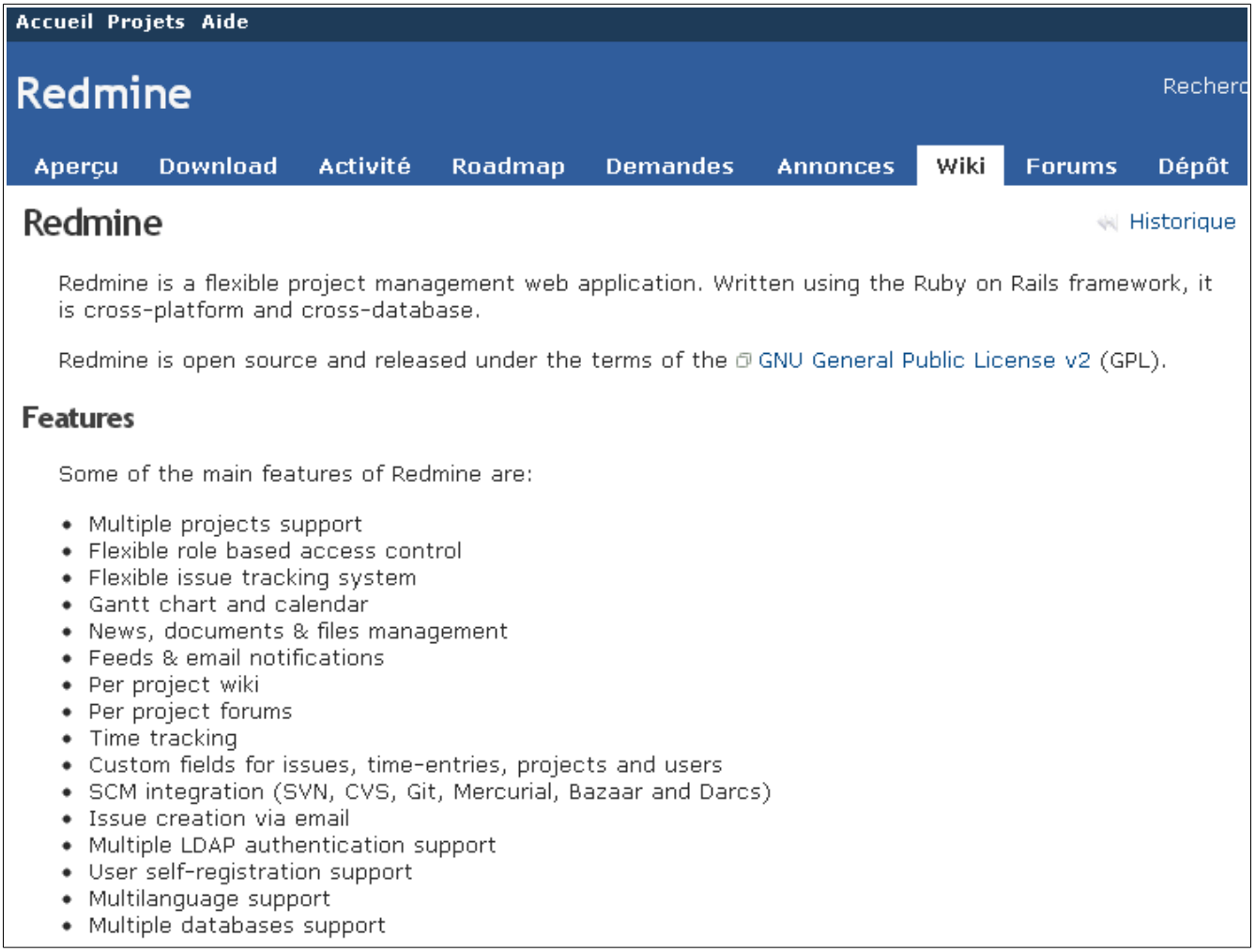
1 http://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions
2 http://fr.wikipedia.org/wiki/Gestion_de_version_d%C3%A9centralis%C3%A9e#Gestion_de_versions_d.C3.A9centralis.C3.A9e
3 http://git-scm.com/docs
4 http://gitlab.org/
5 http://fr.wikipedia.org/wiki/Int%C3%A9gration_continue
6 http://fr.wikipedia.org/wiki/Compilateur

5.3 Système de gestion des projets en mode Web

Comme c'est le cas des projets qui impliquent plusieurs ressources, tâches, livrables, jalons, etc, un système de gestion de projet s'impose. Dans notre cas, la solution doit être orientée développement collaboratif de logiciels, en d'autres termes, il doit permettre un lien avec notre gestionnaire de version déjà choisi dans l'étape précédente et offrir les fonctionnalités de bases suivantes :

- Gestion des droits utilisateurs définis par des rôles;
- Wiki;
- Forums multi-projets;
- Rapports de bogues (bugs), demandes d'évolutions;
- Gestion de feuilles de route.

Le logiciel retenu est **Redmine**[7]**.**



7 http://www.redmine.org/projects/redmine/wiki/Features

5.4 Gestionnaire de dépôt de librairie

L'outil choisi est en lien direct avec le langage de programmation avec lequel les projets sont développés. Les commiteurs décident du logiciel adapté à leurs besoins, en concertation avec le comité technique, ce dernier met à leur disposition cet outil.

5.5 Système d'authentification

Un système d'authentification doit être installé afin de permettre une meilleure sécurité des droits d'accès. Ce système repose sur le protocole LDAP « ***Lightweight Directory Access Protocol*** ».

Le logiciel retenu est 389 directory server.

5.6 Gestionnaire de listes de diffusion[8]

C'est une solution qui permet de tenir informées les différentes parties prenantes d'un projet par le publipostage d'information en utilisant le courrier électronique.
L'outil retenu est Sympa.

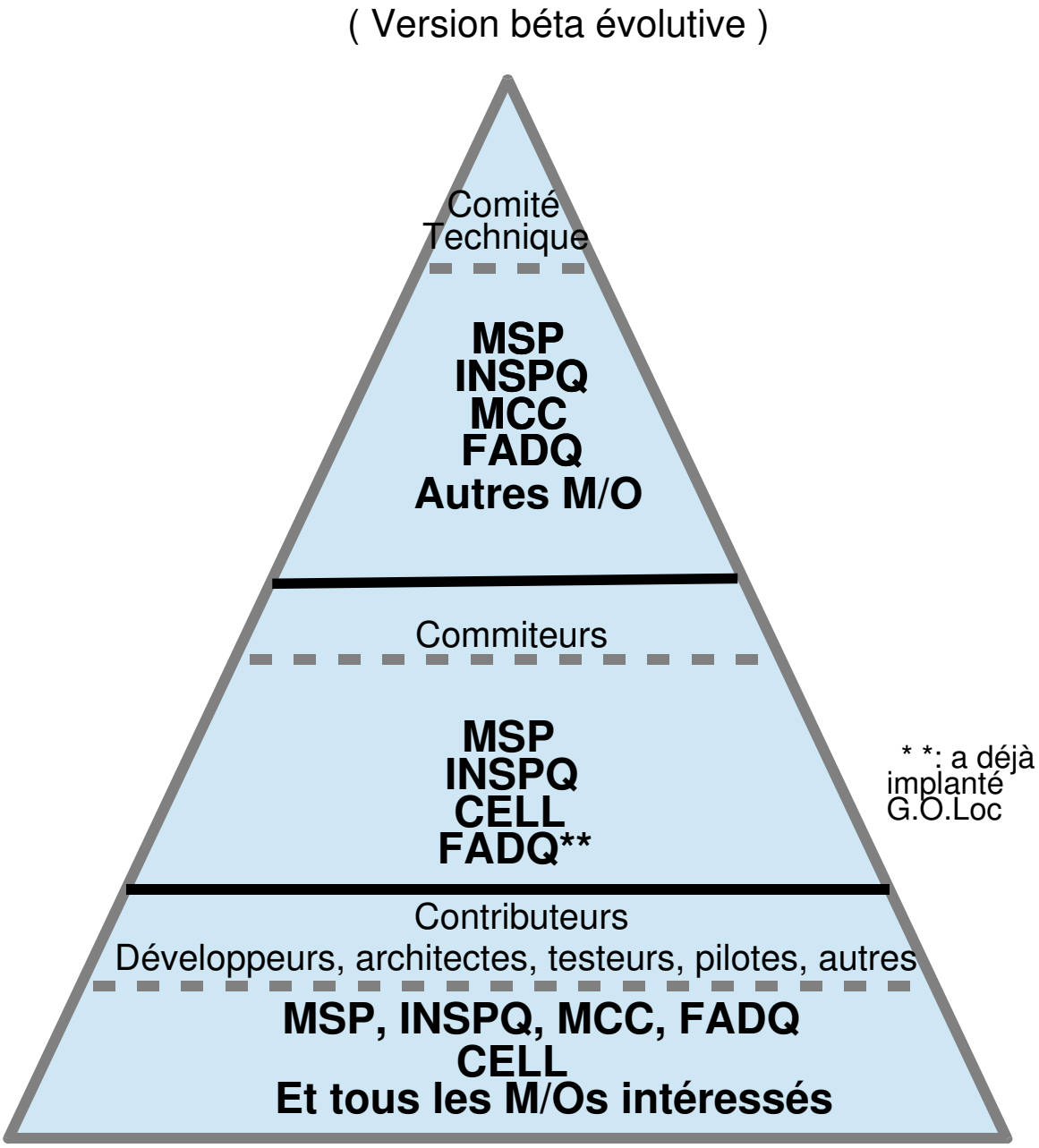
8 http://fr.wikipedia.org/wiki/Listes_de_diffusion

6 Communauté pilote au gouvernement : Communauté Géo G.o.Loc

Un projet géomatique en logiciel libre en sécurité civile du Ministère de Sécurité publique (MSP) et de ses partenaires servira de projet pilote. La première communauté gouvernementale sera constituée autour de ce projet. Il permettra de tester ce modèle de fonctionnement et de le réajuster en fonction des défis auxquels cette communauté sera confrontée.

En **annexe 1**, on retrouve l'architecture de la forge au MSP.

6.1 Acteurs de la communauté du projet Géo G.o.Loc



6.1.1 Comité technique

MSP : Luc Lessard, Nicolas Gignac, Stéphane Cyr
INSPQ : Steve Toutant
MCC : François Gourdeau
FADQ : Patrick Mayrand
Autre M/O : à déterminer

6.1.2 Commiteurs

MSP : Nicolas Brisebois, Marc-André Trottier, Nicolas Gignac
INSPQ : Steve Toutant
CELL : architecte organique à déterminer
FADQ** : Michael Lane **: a déjà Implanté G.O.Loc

6.1.3 Contributeurs

Développeurs, architectes, testeurs, pilotes, autres

MSP : Luc Lessard, équipe géo, pilotes
INSPQ : Pilotes
MCC : Simon Noël, pilotes
FADQ : à déterminer
CELL : développeurs
Et tous les M/Os intéressés

Annexe 1 : Projet pilote d'une forge au MSP