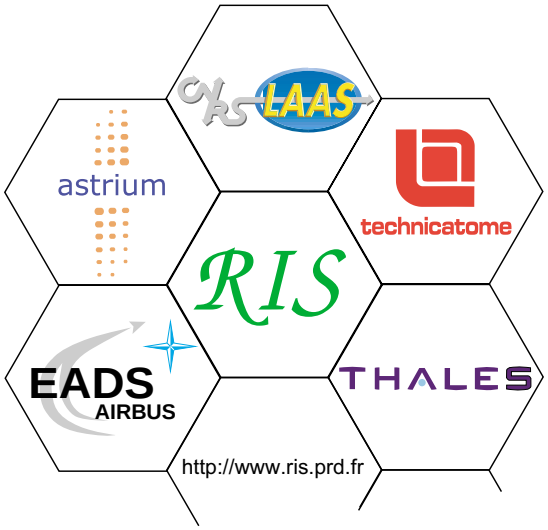

Réseau d'Ingénierie de la Sûreté de fonctionnement

Réunion n°5 du GT Logiciel Libre et Sûreté de Fonctionnement
7 mars 2002 – LAAS-CNRS

Programme	
9 h 00-11 h 15	Introduction
	Le projet OpenBSD : développement d'un système sûr, *Matthieu Herrb, LAAS-CNRS*
	Modèle de développement libre et recherche scientifique: une dynamique autour de B?, *Georges Mariano, INRETS*
11 h 15-11 h 30	*Pause*
11 h 30-12 h30	Logiciels libres: un point de vue commercial, *Cyrille Comar, ACT Europe*
12 h 30-13 h 30	*Déjeuner*
13 h 30-15 h 30	Usage des Logiciels Libres pour la santé – résultats d'un sondage SPIRIT, *Brian Bray, Minoru Development*
	Point de vue du domaine automobile, *Jean-Marc Astruc, Siemens Automotive*
	Mise à jour du planning des prochaines réunions et initiation de la réflexion sur le document final

Participants :

Membres du GT :

B. Bérard (LSV), G. Bulsa (Astrium), Y. Crouzet (LAAS-CNRS), Y. Garnier (SNCF), S. Goiffon (Airbus), G. Mariano (INRETS), V. Nicomette (LAAS-CNRS), J-M. Tanneau (THALES), J-L. Terraillon (ESA), H. Waeselynck (LAAS-CNRS).

Absents ou excusés :

E. Conquet (ESA), P. Coupoux (Technicatome), P. David (Astrium), Y. Paindaveine (Commission Européenne), I. Puaut (IRISA).

Invités :

J-M. Astruc (SIEMENS Automotive), B. Bray (Minoru Development), C. Comar (ACT Europe), M. Herrb (LAAS-CNRS).

1. Introduction

P. David ne pourra pas participer à cette réunion, et transmet ses excuses auprès du Groupe de Travail.

Le Groupe de travail tient à remercier tous les invités qui ont bien voulu apporter leur contribution à la journée. Ainsi, nous avons plusieurs intervenants extérieurs ayant contribué à la création – ou assurant du support sur – des logiciels libres : Brian Bray (Minoru Development Corporation, logiciels libres dans le domaine de la santé), Cyrille Comar (ACT Europe, pour GNAT Ada 95), Matthieu Herrb (LAAS-CNRS, pour OpenBSD). De plus, Jean-Marc Astruc, de Siemens Automotive, a également été ré-invité pour apporter un éclairage "automobile" à la problématique du groupe.

Pour le compte-rendu de la réunion du 14 janvier 2002, H. Waeselynck attend encore quelques retours d'orateurs avant diffusion dans la partie publique du site web du *RIS*.

2. Processus de création et des évolutions des logiciels libres

Le projet OpenBSD : développement d'un système sûr, *Matthieu Herrb, LAAS-CNRS*

OpenBSD est un système UNIX complet, multi-plateformes. D'un point de vue historique, il est affilié au système BSD (Berkeley Software Distribution) développé à l'Université de Berkeley dans les années 80. Après le portage du système sur Intel 386, et l'arrêt des activités BSD à Berkeley, deux communautés ont poursuivi ces développements : la communauté FreeBSD est restée dans le monde PC, tandis que NetBSD s'est orientée vers le multi-plateformes. En 1995, NetBSD a connu une scission, donnant lieu à une nouvelle branche, OpenBSD, fondée par Theo de Raadt. L'accent est mis sur les aspects sécurité (*security*), ce qui se traduit par l'intégration de bibliothèques de sécurité au sein du système, et par le souci de délivrer du code correct. OpenBSD est distribué en libre, sous la licence BSD. Cette licence n'induit pas de contamination des logiciels dérivés, qui peuvent être des logiciels propriétaires pourvu qu'il soit fait état du copyright d'origine.

L'ouverture de OpenBSD à des applications tierces (i.e., hors noyau et applications de base de la distribution) s'effectue au moyen du mécanisme de ports, commun à tous les systèmes BSD. Un port est un Makefile contrôlant le téléchargement, la configuration, la compilation, … de l'application, de façon à rendre son installation très simple (le port doit cependant être à jour vis-à-vis de la version du système). Plus de 1700 applications peuvent ainsi être installées. Une autre possibilité d'ouverture est offerte via l'émulation de différents systèmes UNIX – notamment LINUX – ce qui permet d'exécuter les binaires d'applications non natives.

OpenBSD se veut le système UNIX le plus sûr (*secure*), d'où une démarche de sécurité volontariste (*proactive*) :

- Intégration de mécanismes de sécurité dans le système, pour la connexion à distance, l'authentification, le chiffrement/déchiffrement, et la génération de nombres aléatoires. L'intégration de bibliothèques de cryptographie est rendu possible par la législation du Canada, où le projet est basé. Open BSD bénéficie également de mécanismes hérités de BSD. Par exemple, les niveaux de sécurité sont définis de telle sorte que même le super utilisateur ne puisse pas écrire dans la zone mémoire du noyau.
- Principe "sûr par défault". Ce principe induit une logique d'installation différente de celle classiquement mise en œuvre dans le monde UNIX. Seuls les services et démons indispensables sont actifs par défaut, les autres devant être activés explicitement (à l'opposé de la logique classique, "rien à configurer", où tout est actif par défaut). Un résultat remarquable de cette politique est "quatre ans sans trou de sécurité à distance dans l'installation à défaut ".
- Règles de programmation sûre. Il s'agit de minimiser la taille du code critique du point de vue de la sécurité, et de minimiser les privilèges. Ainsi, si une application nécessite d'effectuer des traitements avec les droits du super utilisateur, on préfèrera confiner ces traitements dans un petit programme externe, plutôt que positionner le bit `setuid` de l'ensemble de l'application. D'autres règles sont la simplicité du code (*K.I.S.S. = Keep It Simple and Stupid*), et l'explicitation des traitements des entrées invalides ou absurdes. Il n'y a pas de manuel de ces règles, mais les développeurs sont sensibilisés par des exposés réguliers.

- Audits du code. Il est bien connu que les “bugs” peuvent induire des comportements exploitables par des attaquants. Un principe de base est de chercher les bugs pour les corriger systématiquement, qu’ils constituent ou non des vulnérabilités pour le système. Pour cela, des audits de code sont régulièrement effectués (lecture attentive, avec recherches sous éditeurs de texte). Bien que non formalisés, ces audits suivent un processus général : lorsqu’un bug est trouvé, on cherche à le généraliser par une classe de fautes ; puis on poursuit l’analyse en essayant de trouver d’autres fautes de cette classe dans le système. Par exemple, un avis du CERT avait signalé une vulnérabilité du serveur FTP de plusieurs systèmes UNIX, due à l’omission, lors de l’appel d’une fonction (`setproctitle`), d’un paramètre de format de chaîne de caractère. Suite à la correction de ce bug dans OpenBSD, un audit du code a permis d’identifier et de corriger plusieurs dizaines d’autres bugs relevant de ce type d’omission.
- Tests. Il ne suffit pas de vérifier le comportement nominal, car les attaquants procèdent souvent en injectant des données absurdes. Des tests de robustesse sont donc effectués, notamment avec des entrées aléatoires. Certains tests sont inclus dans la distribution, et peuvent être rejoués par une commande `make`.

Grâce à cette démarche, on constate que la plupart des avis de sécurité ne concernent pas OpenBSD. Pour renforcer les aspects validation, la communauté serait demandeuse d’outils libres, et de spécialistes disposés à donner de leur temps.

Si la licence BSD n’impose pas de restrictions d’utilisation et de redistribution, il pourrait y avoir des problèmes dus à des logiciels inclus. Un audit est en cours depuis un peu plus d’un an, pour identifier l’origine et la licence de tous les logiciels inclus. Lorsqu’un problème est détecté, on contacte l’auteur d’origine, pour lui demander de modifier sa licence. Si aucun accord à l’amiable ne peut être trouvé, le logiciel en cause est re-développé. Cela a été le cas pour le protocole SSH pour les connexions distantes (la nouvelle version libre, OpenSSH, représente actuellement plus de 50% des implémentations du protocole, d’après une étude de l’Université du Michigan), et le filtre de paquets IPF. Actuellement, il reste à résoudre 1/3 des problèmes identifiés par l'audit. La communauté mène également une action de fond pour la promotion du logiciel libre dans Internet, notamment auprès de l’IETF (Internet Engineering Task Force).

Le projet Open BSD ne reçoit pas de financement institutionnel récurrent. Ponctuellement, il peut recevoir des bourses DARPA ou USENIX pour financer quelques développements. Il bénéficie de dons, et du support de quelques entreprises et laboratoires (temps ingénieurs, matériel). Enfin, une autre source de revenus provient de la vente de CDs, T-Shirts. Le poste principal de dépenses est la connexion réseau aux serveurs du projet, tous les contributeurs (développement, test, documentation) étant bénévoles.

Il y a aujourd’hui une centaine de développeurs OpenBSD enregistrés, dont une quarantaine sont réellement actifs à un moment donné. L’intégration de nouveaux développeurs s’effectue par cooptation. Le fonctionnement est non hiérarchique. Des rencontres régulières, les *Hackatons*, sont organisées à l’occasion d’événements externes (ex :conférences USENIX). En dehors de ces rencontres “physiques”, la coordination des contributions s’effectue via une liste E-mail privée, et un forum en-ligne ICB (Internet Citizen’s Band) d’accès semi-public. Il existe aussi des listes publiques pour les utilisateurs. La distribution est gérée sous CVS, avec accès en écriture pour les développeurs, et accès en lecture anonyme pour tous. Il y a une nouvelle release tous les six mois. L’ajout de nouvelles fonctionnalités est gelé un mois avant la date de la release, pour permettre de valider le code. Ce gel oblige à effectuer des compromis entre le souhait de fonctionnalités supplémentaires, et leur maturité effective. De nombreux projets d’extension de OpenBSD sont en cours, dont, à moyen terme, des extensions temps-réel basées sur RTMX.

Modèle de développement libre et recherche scientifique : une dynamique autour de B?

Georges Mariano, INRETS

L’exposé porte sur l’émergence de développements libres dans le cadre de recherches autour de la méthode B. La méthode B est une méthode formelle de développement de logiciel. Elle procède par raffinements successifs d’une spécification jusqu’à obtention d’un modèle suffisamment concret pour pouvoir générer du code (C ou Ada). Chaque étape de ce processus donne lieu à des obligations de preuve. Les fondements mathématiques de la méthode sont relativement simples, ce qui a facilité son transfert dans l’industrie. Notamment, B est utilisée dans le domaine ferroviaire (logiciels critiques), ainsi que dans le domaine des cartes à puce intelligentes (projets de R&D).

Il existait des outils support dès le début du transfert de B dans l'industrie, et ces outils ont pu être améliorés par confrontation à des projets opérationnels. Mais les ateliers existants sont des logiciels commerciaux, dont la technologie est "verrouillée". A titre d'exemple, le noyau de ces ateliers consiste en un interpréteur du *langage des théories*, un langage spécifique dont il n'existe pas de description dans le domaine public. Les fichiers d'entrée et de sortie des différents outils implémentés au-dessus du noyau (par exemple, l'outil de génération des obligations de preuve) sont sous des formats spécifiques, dont il n'existe pas non plus de description dans le domaine public. Ceci constitue un frein pour les recherches menées autour de la méthode B, qui auraient besoin de mettre en œuvre les idées émergentes au sein de prototypes de démonstration. Il en résulte une dynamique pauvre de la communauté B, et une grande difficulté de transfert technologique des résultats de recherche.

Le problème du verrouillage technologique s'est notamment posé à l'INRETS, il y a quelques années, dans le cadre de recherches sur des métriques pour les sources B. Ces recherches ont nécessité le développement d'un parseur qui puisse se connecter à d'autres outils de traitement. Ceci a amorcé une dynamique pour le développement d'outils libres, selon les grands principes suivants : aucun format propriétaire ; pour chaque outil, utilisation du langage le plus adéquat ; traçabilité entre spécification et outil (ex : code commenté avec des extraits du B Book, l'ouvrage de référence de la méthode B). Le langage choisi pour le parseur a été OCaml, un langage fonctionnel de haut niveau développé à l'INRIA et distribué en libre. D'autres outils ont ensuite été développés (par exemple, un type-checker, un générateur d'obligation de preuves, …), formant la plate-forme *BCaml*. Le langage OCaml est très sophistiqué (il permet par exemple de paramétrer un type-checker par une théorie des types), mais d'usage "pointu". Pour permettre l'ouverture à d'autres technologies (outils B en Java, ..), il a été décidé d'utiliser XML comme format d'échange au centre de la plate-forme. Cela permet de récupérer les technologies déjà existantes autour de XML pour tous les traitements non spécifiques : par exemple, utilisation de la technologie XSLT pour spécifier des règles de transformation des éléments XML, définir des feuilles de styles html, LaTeX, …

En parallèle à la démarche BCaml menée à l'INRETS, d'autres outils B libres ont également émergé. L'idée est maintenant de se fédérer. Ainsi, BCaml est devenu un sous-projet d'une plate-forme élargie, BRILLANT, hébergée sur le site Savannah. En plus des échanges "virtuels" entre contributeurs, des réunions BRILLANT sont organisées tous les six mois. En reprenant la grille d'évaluation RNTL pour les projets logiciels libres, on peut faire les remarques suivantes. BRILLANT répond à un besoin partagé, qui concerne le développement de prototypes de recherche autour de la méthode B. Une communauté est en train de se structurer, mais cela prend du temps. Un problème rencontré est notamment celui du manque de reconnaissance, par les instances d'évaluation académiques, des logiciels libres en tant que résultats de recherche. Au delà de la phase d'émergence, il y a encore un point d'interrogation sur le décollage effectif de BRILLANT.

Logiciels libres: un point de vue commercial, *Cyrille Comar, ACT Europe*

Ada Core Technologies a été fondée en 1994-95, à l'issue d'un projet du DoD sur le développement d'un compilateur Ada 95 (projet GNAT = GNU NYU Ada 9X Translator). L'équipe GNAT s'est constituée en entreprise, et Ada Core Technologies représente actuellement 40-50 personnes réparties entre les USA et l'Europe. Ses activités sont centrées sur Ada, son produit phare étant l'environnement de développement *GNAT Pro*. Il existe également une variante de cet environnement destinée au développement d'applications critiques, *GNAT Pro High-Integrity Edition.* Dans ce dernier cas, les aspects dynamiques sont fortement restreints : on peut par exemple imposer que le code généré n'effectue pas d'appel au run-time Ada, ou se limiter à une librairie run-time très simple fournie dans l'environnement. Tous les produits de ACT sont des logiciels libres. L'offre commerciale met l'accent sur le support technique, avec des engagements forts sur la qualité de service (notamment, temps de réponse aux questions des utilisateurs).

Avant d'expliquer cette stratégie commerciale, C. Comar effectue une mise au point terminologique. La notion de "logiciel libre" diffère de celle de "logiciel du domaine public". Dans le cas d'un logiciel du domaine public, il n'y a plus de détenteur de copyright. Si des logiciels dérivés sont créés, ils ne sont pas du domaine public (sauf démarche explicite des auteurs de ces logiciels dérivés). Dans le cas d'un logiciel libre, il y a un détenteur de copyright, qui place une licence autorisant la copie pour usage personnel, ainsi que la redistribution du logiciel d'origine et de logiciels dérivés. Un re-distributeur n'a pas la possibilité de modifier les termes de la licence, et souvent ceux-ci se propagent aux logiciels dérivés (ex : GPL), qui ne peuvent donc être distribués que sous forme de logiciels libres. Un logiciel libre est toujours open-source. Par contre, les logiciels open-source ne sont pas nécessairement des logiciels libres. Par exemple la licence APL (Apple

Community Licence), attachée à des sources distribués par Apple, n'est pas considérée comme une licence libre, car elle spécifie qu'Apple sera propriétaire de toute modification éventuelle.

La licence n'affecte pas l'auteur d'origine, qui peut choisir de distribuer un même logiciel sous différentes licences. Par exemple, la société Cygnus (actuellement Red Hat) distribuait le produit Cygwin sous deux formes : une version publique sous GPL, et une version payante sous une licence adaptée au développement d'applications propriétaires. Pour sa part, ACT a choisi de proposer une seule distribution, qui garantit l'absence de contamination des développements Ada des utilisateurs. Les sources de la distribution sont soit sous GPL, soit sous GMGPL (GNAT Modified GPL) pour les parties où la GPL poserait des problèmes de contamination (librairies runtime). Notons qu'une autre variante de la GPL, la LGPL, serait moins satisfaisante de ce point de vue : elle impose de garder la possibilité de faire l'édition de lien avec les librairies libres, d'où la nécessité d'inclure les codes objet dans la distribution.

La distribution des sources sous licence GPL ou GMGPL peut sembler une stratégie discutable. En théorie, rien n'empêcherait un concurrent de redistribuer les produits de ACT pour son compte (en pratique cela ne s'est jamais produit). Ou encore, les clients pourraient décider de se passer de support. Dans les faits, une stratégie commerciale basée sur 1) la mise à disposition des sources et 2) un aspect support technique appuyé, s'avère viable. La mise à disposition des sources est un atout pour la certification. Elle est facteur de confiance pour l'utilisateur, qui a la possibilité de vérifier par lui-même la qualité des produits. Elle permet de garantir l'absence de fonctions cachées. Elle protège l'utilisateur en cas de disparition du fournisseur. Enfin, l'accent mis sur la qualité du support répond aux attentes des utilisateurs industriels : en pratique, ces derniers ne seraient pas prêts à utiliser des logiciels sans support, non seulement pour traiter les problèmes, mais aussi pour accompagner les évolutions technologiques des produits.

Usage des Logiciels Libres pour la santé – résultats d'un sondage SPIRIT, *Brian Bray, Minoru Development*

SPIRIT est un projet IST qui rassemble trois partenaires, Minoru, Conecta et Sistema. Son objectif est de faire avancer plus vite le logiciel libre dans le domaine de la santé : développer la taille et l'activité de la communauté, susciter des projets et développements libres, fournir un lieu de réunion virtuel, identifier les meilleures ressources libres et étudier les solutions pour leur diffusion commerciale.

Une des actions du projet a été de réaliser un sondage sur l'utilisation de logiciels libres pour la santé. Un questionnaire de deux pages a été envoyé à un échantillon de 947 personnes ou organismes :

- Des "innovateurs", c'est-à-dire des personnes ayant publié des articles informatiques durant ces dix dernières années. 57097 auteurs ont été recensé, parmi lesquels un échantillon aléatoire de 3128 auteurs a été tiré. Après analyse, 465 ont été retenus (européens, domaine médical).
- Des hôpitaux européens.

77 réponses (8%) ont été reçues.

La première question visait à déterminer le taux de pénétration d'un ensemble de produits logiciels, libres ou pas. L'échelle proposée allait de "jamais entendu parler" à "vital pour notre fonctionnement". Les logiciels les plus utilisés et ressentis comme vitaux sont : Windows NT, Unix (toutes versions confondues), Oracle, Microsoft SQL server, et Microsoft Mail / Exchange. Parmi les autres logiciels bien notés, on peut mentionner Linux et Apache. Par contre, les systèmes de la famille BSD sont totalement inconnus, ainsi que des logiciels spécifiques aux soins de santé. Il faut savoir qu'il n'y a pas de standard pour de tels logiciels santé, et que le marché est très divisé : les logiciels utilisés dans un pays ne sont connus que dans ce pays, voire même certains logiciels ne sont connus que des médecins formés dans telle université.

La deuxième question visait à déterminer s'il serait difficile de convaincre les acteurs santé d'effectuer des développements libres. Il apparaît que, parmi les sondés impliqués dans des développements, 63% déclarent qu'ils pourraient envisager une licence libre.

La troisième question visait à identifier les barrières éventuelles à l'utilisation de logiciels libres. Les réponses montrent que, globalement, il n'existe pas de barrière forte : sur une échelle de 0 (= aucune barrière) à 2 (barrière), la réponse la plus élevée a été de 1.1. Parmi les barrières possibles, il y a la crainte du manque de support, et la perception des logiciels libres comme présentant des risques du point de vue la sécurité-confidentialité. Par contre, les sondés pensent que le logiciel libre est fiable, et commercialement viable.

Une analyse globale a également permis de dégager différents profils type de sondés (par exemple, les "tout Microsoft", les "convertis" favorables a priori à n'importe quel logiciel libre, …). Le sondage sera refait dans un an, pour étudier les évolutions.

Les logiciels spécifiques santé peuvent être classés en trois catégories : pour les hôpitaux, pour les médecins généralistes, pour des réseaux (cette dernière catégorie est très récente). Comme indiqué plus haut, le marché est très divisé, avec des PME nationales (logiciels pour les hôpitaux) et de très petites entreprises (logiciels pour généralistes). Il n'y a aucun standard, aucune interopérabilité, ce qui verrouille la technologie et freine le travail coopératif. Seules les entreprises propriétaires peuvent innover, et le rapport de force pour guider les évolutions n'est pas en faveur des utilisateurs. Un autre exemple de problème est le cas d'un logiciel (gestion de base de données) dont la licence stipule que le client ne publiera pas de données de benchmark sur ce logiciel, sauf autorisation du vendeur. Clairement, les logiciels libres constitueraient un moyen pour les utilisateurs de rentrer dans le jeu. De plus, l'ouverture du code source est un principe qui rentrerait bien dans la logique des médecins (accès aux formules des médicaments, habitude de publication de résultats cliniques et revue par des pairs, …). Il est cependant vraisemblable que le développement de logiciels libres spécifiques santé ne pourra pas décoller sans coup de pouce initial (financement institutionnel, regroupement d'acheteurs).

3. Architecture et validation – compléments d'information

Point de vue du domaine automobile, *Jean-Marc Astruc, Siemens Automotive*

L'électronique embarquée dans l'automobile vise à satisfaire des réglementations de plus en plus sévères (réduction des émissions polluantes, réduction de la consommation de carburant), et à satisfaire les utilisateurs (augmentation des performances / coût, amélioration du confort et des prestations, augmentation de la sécurité). La caractéristique "production de masse", couplée à une forte concurrence, induit une pression sur les coûts, ainsi qu'une course à l'innovation pour se distinguer. Ainsi, des fonctions mécaniques sont remplacées par de l'électronique, et des fonctions type "entertainment" viennent s'ajouter. Il en résulte une complexité et un niveau d'intégration croissants. A titre d'exemple, sur la Laguna, on trouvait 20 calculateurs, chacun dédié à une fonction. Sur la Laguna II, pour le même nombre de calculateurs, il y a trois fois plus de fonctions pilotées par électronique. Les architectures doivent permettre de gérer le réutilisation de fonctions modulaires et adaptatives. Il y a une demande "plug and play" pour pouvoir intégrer facilement des fonctions de différentes origines.

Il n'existe pas de réglementation spécifique pour l'électronique automobile. En matière de sûreté de fonctionnement, c'est le constructeur qui fixe les objectifs. Pour les fonctions critiques (ex : contrôle moteur), il est de plus en plus fréquent que l'appel d'offres impose de suivre le niveau SIL 4 de la norme IEC 61508. En effet, bien que la sûreté de fonctionnement ne soit qu'un des paramètres pris en compte par le constructeur, il y a l'épée de Damoclès d'éventuelles campagnes de rappel, des implications juridiques, et de la sanction des clients. A noter que le domaine automobile pose aussi des problèmes facteurs humains, car l'utilisateur final du système est "n'importe qui", non spécialiste. De plus, on ne peut pas supposer que le système sera maintenu selon des règles prédéfinies. Enfin, l'évolution prévue vers le "tuning" par électronique, c'est-à-dire la calibration du véhicule (nouvelles options acquises, …) après achat, posera de nouveaux problèmes vis-à-vis de la sécurité, notamment en ce qui concerne le tuning pirate.

Pour l'instant, la question de l'utilisation de composants logiciels sur étagères (COTS, LL) ne s'est pas encore posée. Pour les fonctions critiques, ce ne sera probablement pas envisagé avant cinq ans (on dispose d'exécutifs temps réel maison qui répondent aux besoins actuels, et assurent un bon niveau de confiance). Par contre, pour des fonctions de type entertainment, la question pourrait arriver plus tôt.

4. Initiation de la réflexion sur le document final et prochaines réunions

P. David et H. Waeselynck ont préparé une proposition de plan pour le document final (voir site web privé du GT), qui est présentée en séance. L'idée est que chacun se situe par rapport à ce plan, de façon à pouvoir mettre des noms en face de chaque chapitre, à l'issue de la réunion 6.

La réunion 6 sera également consacrée aux thèmes suivants :

- logiciel libre et aspects juridiques, sous forme de table ronde avec des représentants compétents des différents partenaires du *RIS*. Bernard Lang, de l'AFUL, sera également invité.
- Logiciel libre et certification, avec une intervention de Y. Garnier et G. Mariano. J-L. Delamaide, du CEAT, sera également invité pour un point de vue du domaine avionique.

A partir de la réunion 7, les travaux du groupe se concentreront sur la production du document final, en gardant quelques exposés techniques sur des thèmes encore insuffisamment couverts (ex : sécurité informatique).