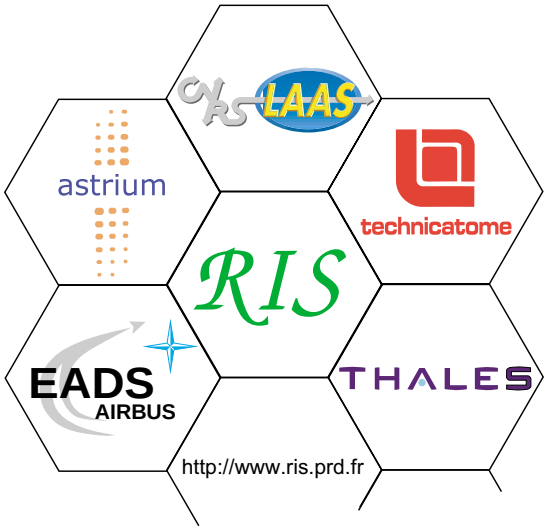


# Réseau d'Ingénierie de la Sûreté de fonctionnement

## Réunion n°2 du GT Logiciel Libre et Sûreté de Fonctionnement
## 11 juillet 2001 – LAAS-CNRS

| Programme | |
|---|---|
| 10 h00-12 h00 | Introduction (approbation du compte-rendu précédent)<br>Terminologie, types de licences<br>Point de THALES sur les aspects juridiques |
| 12 h 00-13 h 15 | *Déjeuner* |
| 13 h15-14 h15 | Positionnement de EADS-AIRBUS et IRISA par rapport à la thématique du Groupe (5mn par intervention)<br>Identification des priorités du Groupe, par synthèse des attentes/craintes exprimées dans l'ensemble des interventions |
| 14 h45-15 h15 | Catégorisation des logiciels libres intéressant le Groupe |
| 15 h 15-15 h 30 | *Pause* |
| 15 h30-16 h30 | Mise à jour du planning des prochaines réunions, et identification des contributions |

**Participants** :

Membres du GT :

B. Bérard (LSV), P. Coupoux (Technicatome), Y. Crouzet (LAAS-CNRS), P. David (Astrium), Y. Garnier (SNCF), S. Goiffon (EADS Airbus), G. Mariano (INRETS), V. Nicomette (LAAS-CNRS), I. Puaut (IRISA), J-M. Tanneau (THALES), J-L. Terraillon (ESA), H. Waeselynck (LAAS-CNRS).

Absents ou excusés :

E. Conquet (Astrium), C. Desquilbet (CEAT), Y. Paindaveine (Commission Européenne).

## 1. Introduction

Le compte-rendu de la réunion du 31 mai 2001 est approuvé. Il pourra être mis dans la partie publique du site web du *RIS*.

H. Waeselynck demandera l'adhésion du GT à la liste électronique *freesw* du groupe de travail européen sur les logiciels libres, créé à l'initiative de la Direction Générale IS (cf. http://eu.conecta.it/).

## 2. Terminologie, types de licences, sigles divers

**Groupes à l'origine du mouvement**

Deux groupes sont à l'origine du mouvement du logiciel libre :

- Richard Stallmann, anciennement programmeur au MIT, a fondé la Free Software Foundation (FSF, http://www.fsf.org), une association à but non lucratif qui pilote le projet GNU (Gnu's Not Unix) et récolte des dons pour le développement de logiciels libres.
- Le deuxième groupe est issu des travaux menés par le Computer Science Research Group à University of California at Berkeley. Il s'est constitué autour du développement de BSD Unix (où BSD est l'acronyme pour Berkeley Software Distribution).

**Logiciel libre (open source software)**

Définition reprise de la Free Software Foundation :

*L'expression « Logiciel libre » fait référence à la liberté pour les utilisateurs d'exécuter, de copier, de distribuer, d'étudier, de modifier et d'améliorer le logiciel. Plus précisément, elle fait référence à quatre types de liberté pour l'utilisateur du logiciel :*

- *La liberté d'exécuter le programme, pour tous les usages (liberté 0).*
- *La liberté d'étudier le fonctionnement du programme, et de l'adapter à vos besoins (liberté 1). Pour ceci l'accès au code source est une condition requise.*
- *La liberté de redistribuer des copies, donc d'aider votre voisin, (liberté 2).*
- *La liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté (liberté 3). Pour ceci l'accès au code source est une condition requise.*

*Un programme est un logiciel libre si les utilisateurs ont toutes ces libertés.*

A noter que les valeurs défendues par le mouvement du logiciel libre peuvent trouver d'autres domaines d'application que le développement du logiciel. On trouvera ainsi les expressions *open hardware* (voir le portail http://opencollector.org), *open law* (http://eon.law.harvard.edu/openlaw/) ou encore *open content* (par exemple, le livre "Logiciels libres : Liberté, égalité, business", par Jean-Paul Smets-Solanes et Benoît Faucon, avait été mis en *open content*, et chacun pouvait proposer des modifications).

**Licences**

L'OSI (Open Source Initiative, http://www.opensource.org/) est un organisme créé pour pouvoir contrôler l'appellation "open source software". Il délivre le label **OSI certified** à des logiciels distribués sous une licence respectant un ensemble de règles formalisé dans le Open Source Definition. Parmi les licences approuvées par l'OSI, on trouve :

- La licence GPL (General Public License) proposée par la FSF, et sa variante LGPL (Lesser General Public License) permettant l'intégration avec des logiciels propriétaires (ex : la GNU C Library est distribuée sous licence LGPL) ;
- La licence BSD issue du groupe de Berkeley.

Leur texte peut être chargé à partir du site de l'OSI.

**Copyright/left**

Le copyright identifie l'auteur du logiciel libre, c'est-à-dire la personne qui lui a attaché une licence. L'idée est de déclarer qu'un logiciel est sous copyright, pour pouvoir lui attacher des conditions de distribution auxquelles il sera interdit d'apporter des restrictions. Le jeu de mot *copyleft* traduit cet esprit : au contraire des licences classiques, les licences des logiciels libres visent à garantir des libertés pour les utilisateurs.

**Quelques projets/produits issus du mouvement des logiciels libres**

GNU – A l'origine, le projet GNU regroupait un ensemble cohérent de développements pour construire un environnement Unix libre. De nos jours, cette cohérence est peut-être discutable. A noter que le label GNU, donné aux développements pilotés par la FSF, n'est pas un label de qualité : il traduit plutôt une philosophie commune. Des recommandations de programmation sont cependant fournies. Il serait intéressant de regarder s'il existe un document donnant une vue d'ensemble de l'architecture Unix sous-tendant le projet GNU.

Gnome, KDE – Ce sont des environnements de bureau pour stations Unix. Historiquement, KDE (K Desktop Environment) était basé sur Qt, un logiciel propriétaire de la société Trolltech permettant le développement d'interfaces graphiques (notamment sous MS Windows). En réaction, la FSF a lancé le projet Gnome (GNU Network Object Model Environment). Trolltech a fini par céder à la pression, et une version libre de Qt est maintenant distribuée, levant ainsi les problèmes attachés à KDE. Cependant, il semble que KDE ait moins réussi que Gnome à fédérer une communauté, probablement du fait de l'attrait du label GNU dans le monde des logiciels libres.

GNAT– Le projet GNAT (GNU NYU Ada 9X Translator) a été mené à l'université de New York, avec un financement du DoD. Il a conduit au développement d'un compilateur libre pour Ada95. Fondée par les auteurs de ce compilateur, la société ACT (Ada Core Technologies) vend du support autour d'un environnement Ada, GNAT Pro. Ce produit est distribué en open-source, sous une licence type GPL mais avec des modifications autorisant le développement d'applications propriétaires. Le support technique est, lui, payant. On constate une qualité de service remarquable : réponse rapide, efficacité pour résoudre les problèmes. En Europe, le support de GNAT Pro est assuré par la société ACT Europe sise à Paris. Si on compare avec un produit concurrent propriétaire, Aonix : la gamme GNAT est plus étendue, mais Aonix s'est spécialisé avec succès dans quelques niches (relatives à des domaines critiques).

Apache – Il s'agit d'un serveur web libre dont le développement est piloté par l'Apache Software Foundation. L'organisation des projets Apache semble très verticale, avec un petit noyau fermé de développeurs. Les interactions avec la communauté sont faibles : les utilisateurs ne contribuent pas aux modifications du code source.

Mozilla – il s'agit d'un navigateur open-source de Netscape, distribué sous une licence approuvée par l'OSI (la licence MPL, ou Mozilla Public License).

**Infrastructure web**

Un *webring* est une structure informelle créant une chaîne de liens entre sites présentant des centres d'intérêt voisins. Chaque site insère sur sa page d'accueil un lien vers le site précédent, un autre vers le suivant et un lien vers la page descriptive du webring. Le terme de webring n'est pas propre à la communauté des logiciels libres.

Les outils de type forums de discussion et FAQ sont des vecteurs importants qui ont favorisé et accéléré la percée des logiciels libres. Ces moyens permettent en effet l'appropriation d'un logiciel par une communauté différente de celle des développeurs : les utilisateurs.

Le site SourceForge (http://sourceforge.net) est une pépinière de logiciels libres, qui offre une infrastructure et des services pour les projets : hébergement gratuit de projets, environnements de développement, forums de discussions,... A ce jour, on compte plus de 26000 projets hébergés, et environ 245000 utilisateurs des services du site. SourceForge a été lancé par la société VA Linux Systems, qui maintient également le portail OSDN (Open Source Development Network).

Le site Savannah (http://savannah.gnu.org), issu de la communauté GNU, est un autre exemple de pépinière de logiciels libres.

Il serait intéressant de savoir si des infrastructures similaires existent en Europe.

## 3. Point de THALES sur les aspects juridiques

Jean-michel Tanneau reprend l'exposé qui avait été présenté par THALES lors de l'Atelier Thématique du 14 décembre 2000 (voir compte-rendu correspondant), en apportant des précisions complémentaires.

**Risques attachés aux licences des logiciels libres**

Un risque évident concerne la propagation des termes de la licence. A ce titre, la licence GPL est dangereuse, puisqu'elle se propage à toute application intégrant tout ou partie d'un logiciel libre. Elle stipule cependant (Article 2 de la traduction http://www.linux-france.org/article/these/gpl.html) : "*Si des éléments identifiables de ce travail ne sont pas dérivés du Programme et peuvent être raisonnablement considérés comme indépendants, la présente Licence ne s'applique pas à ces éléments lorsque Vous les distribuez seuls.*" Malheureusement, le fait de pouvoir "*être raisonnablement considérés comme indépendants*" est sujet à interprétation. L'interprétation de THALES est la suivante : tout code intégrant du code GPL devient GPL ; par contre, un code qui, à l'exécution, charge dynamiquement une librairie GPL, reste propriétaire (il peut être distribué indépendamment de la librairie).

A noter qu'il n'y a jamais eu de poursuites judiciaires à l'encontre de sociétés ayant violé la GPL. En fait, il n'est pas sûr que cette licence tienne devant des tribunaux, du fait des imprécisions que comporte sa rédaction : voir exemple plus haut, ou encore dans l'Article 3 (souligné par nous) "*Toutefois, l'environnement **standard** de développement du système d'exploitation mis en oeuvre (source ou binaire) -- compilateurs, bibliothèques, noyau**, etc**. -- constitue une exception*". Une analyse des failles dans la rédaction de la GPL peut être trouvée dans un article intitulé "Opinion : GPL, dangereuse GPL", que Jean-Michel Tanneau nous communiquera pour diffusion sur le site web privé du GT[1].

Un autre risque est la présence cachée d'algorithmes protégés par brevet. Un exemple bien connu est l'algorithme de compression LZW pour générer des fichiers GIFs, sous brevet UNISYS. Généralement, le copyrighter du logiciel libre désengage sa responsabilité, si bien qu'il incombe à l'utilisateur de vérifier l'absence d'éléments protégés. Ce problème est particulièrement difficile à traiter. Lorsqu'un logiciel libre incluant des éléments protégés est utilisé dans un cadre commercial, les conséquences peuvent être néfastes à deux titres : d'une part, il faudra payer des royalties au détenteur du brevet ; d'autre part, le client pourra éventuellement exiger une nouvelle version du produit, refaite sans ces éléments protégés.

Il semblerait que la FSF ait une démarche pour vérifier l'absence d'éléments protégés dans la distribution GNU. Cependant, aucune garantie n'est offerte. Par ailleurs, une certaine partie de la communauté du logiciel libre mène des actions contre la brevetabilité du logiciel. Pour sa part, la position de THALES est de promouvoir la brevetabilité du logiciel en tant que tel et, à ce titre préconise le retrait de l'article 52 - § 2 & 3 du CBE (Convention sur le Brevet Européen), de même pour les articles L.611-10 (2) et (3) du Code de la Propriété Intellectuelle; ceux-ci excluant le logiciel des items brevetables. Il est à noter que ces articles ont été contournés (nombreuses jurisprudences) à partir du moment où les 3 conditions : inventif (créatif), nouveau (par rapport à l'état de l'art) et pouvant donner lieu à industrialisation, sont réunies.

**Analyses effectuées par le service juridique de THALES**

Le service juridique de THALES effectue des analyses de licences au fil des demandes de qualification des composants (COTS ou LL). Un souci majeur est la rédaction des conclusions de ces analyses en des termes compréhensibles par les ingénieurs. Ainsi, le service juridique effectue une "traduction" des principaux articles de la licence, puis donne un commentaire résumé sur ce qui est autorisé ou non, et sur la validité juridique des termes de la licence. Ces analyses sont rentrées dans une base de données qui répertorie les composants, leur licence (texte d'origine + analyses), l'affaire dans laquelle ils ont été utilisés,…

Une analyse contextuelle de la licence doit également être faite, selon les caractéristiques d'utilisation du composant. Un exemple de problème contextuel est l'interdiction de lier un logiciel GPL à d'autres logiciels libres distribués sous certaines licences non GPL (voir l'article "dangereuse GPL" cité plus haut). Dans le cadre d'une affaire, il faut vérifier que le contrat passé avec le client n'est pas incompatible avec l'utilisation d'un logiciel libre donné.

**Critères de qualification de logiciels libres**

Outre les aspects précédents, certains critères minimaux doivent être remplis pour pouvoir utiliser des logiciels libres dans le cadre d'affaires.

L'origine du composant doit pouvoir être établie. Un composant sans copyright est systématique écarté. Le téléchargement à partir d'un site miroir n'est pas considéré comme offrant des garanties suffisantes pour l'authentification. En fait, plutôt que le téléchargement, THALES préfère souvent demander l'envoi payant d'un CD, en essayant de s'adresser directement au créateur du logiciel.

[1] Maintenant accessible à l'adresse : http://www.ris.prd.fr/GT/LL/private/Reunion2/dangereuse-GPL.htm

Un autre critère obligatoire est l'existence d'au moins une société assurant le support (maintenance, conseil) du logiciel libre.

**En conclusion**

La démarche de THALES, qui a démarré par une analyse des risques, a débouché sur la mise en place d'un processus formalisé permettant de cadrer l'utilisation de logiciels libres.

Les membres du GT pensent que la consolidation de ce processus, et son extension aux autres domaines industriels et académiques du RIS, pourrait constituer un résultat important du Groupe de Travail.

## 4. Positionnement de AIRBUS et IRISA par rapport à la thématique du Groupe

**Isabelle Puaut (IRISA)**

L'IRISA (Institut de recherche en informatique et systèmes aléatoires) est un laboratoire associé à L'INRIA, au CNRS, à l'Université de Rennes 1 et à l'INSA de Rennes. Les recherches se développent au sein de projets structurés autour de grands thèmes scientifiques. La participation au GT concerne le projet Solidor qui, dans le thème "Réseaux et Systèmes", s'intéresse notamment aux applications temps-réel critiques et à leurs supports d'exécution.

La vérification qu'une application satisfait ses échéances temporelles fait appel à des évaluations de pire cas du temps d'exécution. Pour cela, les travaux en cours portent plus particulièrement sur la mise en oeuvre de techniques d'analyse statique du code source. Ces techniques sont appliquées non seulement au niveau du code applicatif, mais également au niveau du code source du système d'exploitation, afin de pouvoir estimer les pires cas relatifs aux appels système. Des analyses ont ainsi été menées sur un système d'exploitation open-source, le noyau RTEMS.

Une autre contribution du projet Solidor concerne la conception d'une couche middleware fournissant des mécanismes de détection et de confinement d'erreurs au-dessus de systèmes d'exploitation temps-réel. L'efficacité de ces mécanismes est validée expérimentalement par des campagnes d'injection de fautes. La mise en œuvre de l'injection de fautes est facilitée lorsque l'on dispose de l'accès au code source du système d'exploitation, d'où – là encore – un fort intérêt pour les systèmes open-source. Initialement conçue au-dessus du noyau Chorus, la couche middleware a pu être portée sur RTEMS. Son portage sur RTLinux pourrait être envisagé prochainement.

**Serge Goiffon (Airbus)**

Le département logiciel embarqué de EADS Airbus a notamment travaillé sur le développement de l'ATSU (Air Traffic Services Unit), un calculateur qui gère les communications sol/bord et sera utilisé dans le FANS (Future Air Navigation System). Dans le cadre de ce projet, et pour la première fois dans le domaine de l'avionique embarquée, un système d'exploitation COTS (LynxOS) a été choisi. Airbus se pose maintenant la question du remplacement de ce COTS par un système open-source, Linux. Cette question est actuellement à l'étude dans le cadre d'un projet de recherche, EROSS (Etude et Recherche sur Open-Source Software).

Les travaux menés au sein de EROSS ont conduit à la réalisation d'un prototype ATSU utilisant Linux. Un intérêt majeur est l'interopérabilité puisqu'on trouve alors le même système d'exploitation sur système cible, sur baie de validation, et sur plateforme de développement (PC). Divers logiciels libres viennent enrichir les environnements de programmation et de validation : outils GNU pour la compilation, l'édition de lien ou le débuggage, outils IHM pour les simulateurs des équipements sol et bord.

Le portage sur Linux n'a pas nécessité de modification importante des sources : quelques drivers ont dû être re-développés ; le *file system* de Linux a été modifié (par suppression de parties de code) pour éliminer les problèmes d'indéterminismes dus au cache. Actuellement, le prototype ATSU inclut simplement le noyau Linux standard, car l'ATSU ne requiert que du temps-réel "mou". Cependant, diverses solutions ont été étudiées pour obtenir un système d'exploitation temps-réel basé sur Linux (RTLinux, TimeSys Linux, MontaVista). L'adoption de RTLinux, par exemple, nécessiterait des modifications importantes du source applicatif, ce qui ne serait pas le cas pour la solution MontaVista.

Les problèmes ouverts concernent maintenant : l'évolution du processus de développement pour prendre en compte l'intégration de logiciels libres, la certification, et la maîtrise des aspects juridiques.

## 5. Rebouclage sur l'ensemble des interventions

A partir des attentes/craintes exprimées dans l'ensemble des interventions de la 1ère réunion, les animateurs avaient préparé une synthèse sous forme de liste structurée en 5 grande rubriques : aspects économiques, aspects techniques, aspects juridiques, aspects organisationnels, processus et certification. Cette liste a été discutée et amendée en réunion. Ci-dessous la liste, avec les amendements donnés en italique (*ajouts* ou ~~*retraits*~~) ainsi que quelques commentaires.

### Aspects économiques

***Rentabilité ?***

- ☺ Abandon de solutions internes
- ☺ Accès à la technologie plus facile ➪ déverrouillage du marché vers d'autres entreprises
- 😐 Pas d'arrêt de commercialisation et de maintenance (mais coût interne de la maintenance)
- 😐 Migration des coûts = développement produit ➪ coûts d'appropriation de la technologie, coûts d'intégration~~, *QoS*~~
- 😐 Injection de fonds (publics, privés) pour soutenir projets LL

***Concurrence ?***

- 😐 Protection des intérêts privés

*Dans ces aspects, bien distinguer ce qui est relatif aux applications métier, et ce qui est relatif aux autres développements.*

*Il serait intéressant d'avoir un retour sur les coûts d'appropriation de la technologie, d'après les expériences vécues par les membres du GT.*

*Egalement intéressant à regarder : le chiffre d'affaires de la FSF, pour avoir une idée du montant de ce que donnent les sponsors des logiciels libres.*

### Aspects techniques

***Sélection LL ?***

- 😐 Identification des LL candidats : maturité, couverture fonctionnelle des besoins, *interopérabilité*
- 😐 Comparaison objective de plusieurs produits : benchmarking ?
- ☺ ~~*Interopérabilité,*~~ implémentation de standards

***Validation (LL et système incluant LL) ?***

- ☺ Grand nombre d'utilisateurs et de dévermineurs
- 😐 Parfois, tests livrés avec sources
- ☺ Accès au source facilite validation : sur code (analyse statique, test, injection de fautes) ou sur modèle "représentatif" (model-checking)
- 😐 Mais disponibilité d'informations sur la conception ? Spécification a posteriori ?
- 😐 Acquisition d'information sur le comportement du LL, et notamment caractérisation des modes de défaillance
- 😐 Vérification de non-régression quand évolution
- 😐 Validation de choix architecturaux

***Architecture ?***

- ☺ Accès au source facilite intégration : modification source, élimination de code ~~*mort*~~ selon restrictions d'utilisation, *adaptation à la plateforme par compilation conditionnelle*
- 😐 Mise en place de parades / défaillances LL
- 😐 Architectures robustes aux évolutions

*Pour déterminer la maturité d'un LL, on regarde quelle communauté est à l'origine de son développement, qui en sont les utilisateurs, et s'il existe des sociétés qui en assurent le support (cf. critères THALES).*

*Dans le cas de Linux, il existerait un dossier de conception à l'adresse http://www.kernel.org*

*Attention, les modifications du source du LL ont un coût en termes de validation (notamment, analyse d'impact de ces modifications ?)*

**Aspects juridiques**

***Droits attachés aux LL ?***

- 😐 Très variables selon licences
- ☹ Risque de propagation des termes de la licence
- 😐 *Pas de jurisprudence*
- ☹ Eléments protégés par brevet inclus (de façon cachée ?) dans les LL
- ~~😐 *Si nous devions définir une licence LL, qu'aimerions-nous y voir ?*~~

***Responsabilité en cas de dommages dus à LL ?***

- 😐 Pas de jurisprudence
- 😐 Responsabilité de l'intégrateur *(idem COTS)*

**Aspects organisationnels**

***Evolution métiers ?***

- 😐 Développeurs ➪ intégrateurs, ~~experts impliqués~~ *surveillance des évolutions* dans communauté LL

***Procédures et structures chargées de leur mise en œuvre ?***

- 😐 Identification des risques selon licences
- 😐 BdD LL
- 😐 Règles cadrant l'utilisation de LL (selon les différents types d'utilisation)

***Stratégie ?***

- 😐 Formes d'intervention dans le cercle vertueux : influer sur le développement de nouveaux LL ? Sur l'évolution de LL existants ? Sur sa validation ? Retours vers la communauté ?
- 😐 Partenariats industriels vs. influence sur communauté du libre ? Stratégie commune autour de standards ?
- 😐 LL génériques (ex: OS) vs. LL spécifiques à un métier
- 😐 Collaborations autour de plate-formes expérimentales
- 😐 Compétences internes sur LL

*Actuellement, la stratégie adoptée par les membres du GT vis-à-vis de la communauté LL est plus une stratégie d'acquisition que de contribution.*

**Processus et certification**

***Evolution des normes ?***

- 😐 Processus ➪ produit
- 😐 Système ➪ composant (briques de bases certifiées ?)
- 😐 LL pour quels niveaux de criticité ?

***Processus création LL ?***

- 😐 Modèle de développement coopératif ?
- 😐 Collaboration logique "libre" vs. formel de la certification (injection d'outils de validation, specs formelles, BdD bugs ?)

***Processus système ?***

- 😐 LL outil support de développement vs. composant intégré dans un système
- 😐 Authentification de l'origine du LL et procédure d'enregistrement des modifications
- ☺ Gestion des évolutions selon calendrier projet du système incluant des LL
- 😐 Mais problèmes de maintenance : choix de version d'un LL en fonction de ses évolutions, synchronisation des évolutions croisées (système + autres LL)
- 😐 Modification du processus de développement pour prendre en compte les aspects spécifiques, notamment étape de sélection, validation
- 😐 *Non-respect de standards de programmation éventuellement problématique (idem COTS)*
- 😐 Documentation du processus de développement

## 6. Catégorisation des logiciels libres intéressant le Groupe

Le tableau ci-dessous récapitule les catégories de logiciels libres qui peuvent apparaître aux différentes couches d'un système (1ère colonne). Les colonnes suivantes donnent, pour chaque couche logicielle, des exemples d'environnement support (matériel, outils) qui pourraient également être libres, ainsi que les standards implémentés.

| **Logiciel** | **Matériel** | **Outils** | **Standards** |
|---|---|---|---|
| Applications | Equipement | IHM | Standard d'interface utilisateur |
| Middleware | Architecture | Langage de description des interfaces, (de)marchalling | Standard de middleware |
| Protocoles | Standard implanté dans un chip | | Standard de communication |
| Langage | JavaChip | Compilateur | Standard de langage |
| Exécutifs | | Atelier de programmation | Standard d'interface OS |
| BSP, drivers | CPU | Simulateur, outils de test | Jeu d'instruction standardisé |

## 7. Prochaines réunions

Le thème “architecture et validation” étant très riche, il lui sera consacré deux réunions (réunions 3 et 4) plutôt qu’une seule, comme initialement prévu dans le planning (cf. compte-rendu réunion 1).

La réunion 3 aura lieu le 20 septembre. Il y aura des présentations de : IRISA, LAAS, LSV, Technicatome + un exposé court de l'ESA.

Les autres membres du GT doivent réfléchir à des interventions pour la réunion 4, éventuellement en sollicitant des collègues de leur entreprise. P. David et H. Waeselynck essaieront de solliciter une intervention dans le domaine automobile, non couvert par les membres du GT. Un contact sera pris avec Siemens Automotive.