



*R*éseau
d' *I* ngénierie
de la *S* ûreté de fonctionnement

Développement de systèmes critiques intégrant des logiciels libres

Agenda

- **Contexte**
- **État des lieux**
- **Logiciel libre et systèmes critiques**
- **Exemples d'architectures**
 - ✓ RTEMS
 - ✓ Linux
- **Conclusions et perspectives**

Le futur des systèmes critiques

- La digitalisation de la société rend l'utilisation des systèmes critiques plus répandus
- Contraintes de coût de plus en plus sévères.
 - ➔ Privilégier la réutilisation plutôt que de procéder systématiquement à de nouveaux développements
- Interopérabilité des systèmes est nécessaire
 - ➔ Utilisation de standards d'interface (commerciaux ou industriels)
 - ➔ Prise en compte des contraintes de sécurité-confidentialité.
- Emploi croissant de logiciels
 - ✓ 48 ko sur SPOT 1 en 1980 ➔ 1,2 Mo sur Mars Express en 2003.
 - ✓ 25 Ko sur A300B en 1974 ➔ 64 Mo sur A380 en 2005.
- Extension de la certification à de plus en plus de systèmes.

Comment gérer le logiciel?

- **Constat des industriels: les COTS ne représentent pas une solution satisfaisante pour les systèmes critiques**
- **Des compétences spécifiques, fines, sont nécessaires**
- **L'utilisation de logiciels pré-développés est une bonne approche**
 - ✓ Permet de se concentrer sur son domaine de compétence
- **Comment se positionner par rapport aux logiciels libres**
 - ✓ Est-ce une menace?
 - ✓ Est-ce une opportunité
- **Décision du RIS d'effectuer une synthèse sur ce sujet**

État des lieux

➤ Défense des utilisateurs face aux éditeurs de logiciels

- ✓ 1984, Manifeste GNU de Richard Stallman
- ✓ 1985, *Free Software Foundation*
- ✓ 1990, Maturité du projet GNU/Linux
- ✓ 1990, Open Source, terme déposé par l'OSI
- ✓ 2000, Reconnaissance politique et économique

➤ Le principe : contraindre à la liberté (par une licence)

1. Liberté d'utilisation
2. Liberté de modification
3. Liberté de (re)diffusion
4. Liberté d'accès au code

➤ Organisation

- ✓ Structuration des projets Logiciels Libres similaire aux projets industriels
- ✓ Financement des projets par des fondations, des dons, des industriels
- ✓ Reconnaissance de la compétence des développeurs par une hiérarchisation

Logiciel libre (LL) et systèmes critiques

hérésie ou réalité de demain ?

Expériences d'utilisation des COTS

➤ Besoins du système

- ✓ Connaissance détaillée du COTS
- ✓ Adaptabilité du COTS au système
- ✓ Disponibilité du dossier de certification
- ✓ Disponibilité du COTS sur une période de 5-10 ans
- ✓ Maintenance sur le long terme : 10 à 20 ans
- ✓ Respect des standards
- ✓ Coût et contenu de la licence

➤ Inconvénients rencontrés

- ✓ Donner du support peut ne pas être intéressant pour le fournisseur
→ coût du support
- ✓ Le client ne connaît pas le COTS
→ coût du dossier de certification
- ✓ Divergence des intérêts entre l'industriel et le fournisseur sur le suivi du marché
→ coût de gel du COTS
- ✓ Stabilité de la solution dans le temps
→ coût de maintenance
- ✓ Pour des petites séries
→ coût des licences est significatif
- ✓ Les clauses de propriété peuvent être bloquantes
→ négociation des licences

Gestion des risques

➤ Risque lié à la licence COTS

- ✓ Problème stratégique
- ✓ Les industriels sont rompus à cette gestion des règles de protection.

➤ La défaillance du composant

- ✓ Si défaillance, la responsabilité du fournisseur est limitée.
- ✓ Le risque de propagation des erreurs du COTS vers le système est un risque réel qui doit alors être géré par l'industriel.
- ✓ L'industriel n'a pas la connaissance du COTS.
- ✓ La relation de confiance avec le fournisseur est insuffisante à la vue des enjeux pour l'industriel.

➤ Disparition fournisseur ou COTS

➤ Risque lié à la licence LL

- ✓ Liberté d'utilisation
- ✓ La GPL apporte des risques nouveaux (contamination)

➤ La défaillance du composant

- ✓ Utilisateur est seul responsable
- ✓ Le risque de propagation des erreurs doit être géré par l'industriel.
- ✓ L'industriel a le code source
- ✓ L'industriel peut acquérir la technologie

➤ LL maintenu par l'industriel

Impact de la maintenance

- **Les durées de vie des systèmes critiques sont longues**
 - ✓ Satellites: 15 ans
 - ✓ Contrôle commande des chaudières nucléaires des sous-marins : 40 ans
- **La politique de maintenance est un des axes de conception du système**
 - ✓ Principes d'architecture qui minimisent les impacts des changements de version
 - ➔ Cette conception va favoriser les standards d'interface
 - ➔ Des mécanismes d'encapsulation du COTS vont permettre de changer les versions avec un minimum d'impact sur le système.
- **La maintenance long terme doit inclure la gestion du risque fournisseur**
 - ➔ Disponibilité du code source est nécessaire

L'apport des logiciels libres pour le système

- L'effort d'appropriation de la technologie d'un LL permet-il de mieux gérer le développement d'un système critique ?
- Plusieurs scénarios d'utilisation des LL en fonction :
 1. phase d'acquisition de la technologie du logiciel libre.
 2. effort d'adaptation du logiciel libre au système.
 3. mise en place du dossier de certification.
 4. maintenance en phase opérationnelle.
 5. mise en place d'une équipe de maintenance à long terme du logiciel libre.
 6. gestion des évolutions majeures du système.

Scenario d'utilisation d'un LL

| | Acquisition technologique | Adaptation au système | Dossier de certification | Maintenance en opération | Maintenance long terme | Évolutions majeures | Synthèse |
|---|---------------------------|-----------------------|--------------------------|--------------------------|------------------------|---------------------|--|
| Scénario 1 ✓Pas de certification ✓Pas de maintenance | Pas nécessaire | Fournisseur | Non nécessaire | Fournisseur | Code source | Fournisseur | Pas d'investissement. Risque assumé. Cas du spatial aujourd'hui |
| Scénario 2 ✓Pas de certification ✓Maintenance | Fournisseur | Fournisseur | Non nécessaire | Par l'industriel | Par l'industriel | Par l'industriel | Investissement dans une équipe de maintenance LL Maintenance interne si le logiciel est critique (OS) Spatial en évolution |
| Scénario 3 ✓Certification ✓Pas de Maintenance | Fournisseur | Par l'industriel | Par l'industriel | Fournisseur | Fournisseur | Fournisseur | Acquisition technologique. Certification par l'industriel. Pas de maintenance LL |
| Scénario 4 ✓Certification ✓Maintenance | Fournisseur | Par l'industriel | Par l'industriel | Par l'industriel | Par l'industriel | Par l'industriel | Acquisition technologique. Certification par l'industriel. Équipe de maintenance LL |

Avantages apportés par les LL

➤ **Accès au code source**

- ✓ Permet de se prémunir contre les risques d'évolution non maîtrisée du logiciel
- ✓ contre le risque de disparition du fournisseur
- ✓ Risque : utiliser le code source en cas de problème sans le maîtriser, c'est le cas aussi pour les COTS

➤ **Utilisation d'un fournisseur de technologie**

- ✓ Même mode de prestation que pour les COTS
- ✓ Le support fourni est souvent de meilleure qualité que pour un COTS car la raison d'être du fournisseur est très clairement sa maîtrise du logiciel

➤ **Acquisition de la technologie**

- ✓ Acquisition de la connaissance fine du logiciel libre peut être longue et coûteuse (plusieurs homme.année)
- ✓ L'investissement est lourd, sur le court et long terme, car il faut conserver une équipe compétente pendant la durée de vie du système.

Sûreté de fonctionnement des LL

- **Une structure d'accueil architecturale est nécessaire:**
 - ✓ Le logiciel libre est suspect au sens de ses modes de défaillances
 - ✓ Ses fonctionnalités peuvent être surabondantes ou mal adaptées
- **Utilisation d'empaquetage ou *wrapper***
- **Partitionnement du système critique en zones de criticités différentes**
 - ✓ Ces mécanismes de confinement des erreurs autorisent l'ouverture des systèmes critiques vers des fonctions d'interopérabilité.
- **Sécurité-confidentialité**
 - ✓ A prendre en compte car le LL a été développé par un tiers, parfois mal identifié

Certification

- La certification correspond à l'acceptation par un organisme tiers de la preuve que le niveau de sûreté de fonctionnement demandé est atteint.
- La certification a une forte influence sur la conception du système.
- **Sûreté de fonctionnement et capacité à être certifié ne sont pas prises en compte par les LL.**
- ➔ **Friilosité des industriels à adopter les LL**
 - ✓ Responsabilité de l'industriel.
- **Notre objectif : analyser les processus de certification des divers secteurs industriels pour**
 - ✓ identifier les méthodes et efforts à fournir pour rendre le système certifiable
- ➔ **LL doit démontrer un apport compétitif certain pour le système**
- ➔ **Sans introduire de risque supplémentaire**

Certification : vue des divers domaines industriels

- la classification en termes de criticité est très homogène entre les divers secteurs industriels.
 - ✓ DAL (*Development Assurance Level*) dans le domaine aéronautique
 - ✓ ou SIL (*Safety Integrity Level*) dans le domaine ferroviaire

| Catégorie | Ferroviaire | Aéronautique | Espace | Nucléaire |
|------------------------------|-------------|--------------|----------------|-----------|
| Pas d'impact | SIL 0 | E | / | / |
| Impact sur le système | SIL 1-2 | C-D | critique | B et C |
| Impact sur les vies humaines | SIL 3-4 | A-B | catastrophique | A |

Impact des niveaux de sécurité Aviation Civile

- Analyse de sécurité descendante du niveau système jusqu'à tout équipement terminal qui contribue à la sécurité.
- L'autorité de régulation a défini un référentiel permettant de garantir la sécurité du système:
 - ✓ Document ARP 4754 au niveau système
 - ✓ Document DO-178B, pour le développement des logiciels critiques (commandes de vol)
- Différentes catégories de logiciels (de A à E) sont définies en fonction de la gravité des conditions de défaillance au niveau système auxquelles le logiciel est susceptible de contribuer (ARP 4754).

Solutions architecturales en fonction de la criticité

| Classification des conditions de défaillances | Degré de redondance | | |
|---|-----------------------------------|-----------------------------------|-----------------------------------|
| | 0 Simple défaillance/erreur | 1 Double défaillance/erreur | 2 Triple défaillance/erreur |
| Catastrophique | A | B | C |
| Dangereux | B | C | D |
| Majeur | C | D | D |
| Mineur | D | D | D |
| Pas d'effet sur la sécurité | E | E | E |
| DAL logiciel (Development Assurance Level) | | | |

- une fonction critique capable d'entraîner une défaillance catastrophique du système peut :
- Ne pas être redondée. Dans ce cas, elle correspondra à un logiciel de catégorie A;
 - Être redondée. Chacune des versions du logiciel sera classifiée en logiciel de catégorie B;
 - Est tripliquée alors chaque version du logiciel sera classifiée en catégorie C.

Certification - LL- Sûreté de fonctionnement

- Un système critique peut être conçu à partir de fonctions moins critiques redondées, si ces redondances sont gérées selon les exigences de sécurité de niveau système.
- Des protocoles de communication ou des exécutifs temps réel, sont potentiellement utilisables dans les niveaux B ou C.
 - Si redondance alors des LL peuvent être utilisés (niveaux C ou D)
- L'emploi de logiciels libres aux niveaux A ou B correspond à un processus particulier où le logiciel a été développé spécifiquement par l'industriel et certifié comme tel
 - création de LL

Méthodologie de développement

- **Le DO-178B définit des objectifs à atteindre**
 - ✓ Le contenu du dossier de certification et des éléments de preuve à apporter est un travail de négociation entre l'industriel et le certificateur.
 - **Dans les domaines du nucléaire, ferroviaire et spatial, une méthodologie de développement est imposée par catégorie.**
 - **Trois objectifs fondamentaux :**
 - 1) évitement de fautes apporté par la rigueur du développement,
 - 2) élimination des fautes par les tests et les essais,
 - 3) protection vis-à-vis des fautes résiduelles par l'emploi de fonctions dédiées à la tolérance aux fautes et à l'amélioration de la robustesse du logiciel
- ➔ Harmonisation aisée entre ces domaines industriels : cycle de vie et méthodes similaires**

Développement et test : indépendant du système

Les efforts à produire sont de deux ordres :

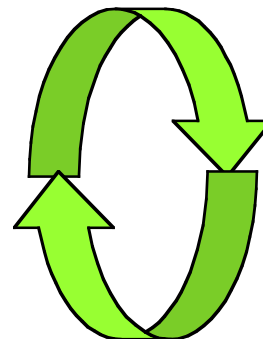
- Des efforts génériques qui ne dépendent que du logiciel et dont les résultats seront exploitables pour tout système hôte qui va utiliser ce logiciel.
 - ✓ Activités de documentation
 - ✓ Tests
- Des efforts spécifiques au système, qui sont orientés principalement vers la sécurité et l'interface avec le matériel.
 - ✓ Le plan d'assurance de la sécurité. Pour des niveaux SIL 1 et SIL 2 (C, D), les exigences de sécurité sont très limitées.
 - ✓ L'intégration avec le matériel.
 - ✦ le test du logiciel sur le matériel est à refaire pour chaque projet.
- ➔ **Les efforts de développement et de test du LL sont en majorité indépendants du système ou à refaire de toute façon.**

Méthodes pour intégrer un LL dans un système critique

- **L'analyse des différentes normes de certification employées dans les secteurs industriels des systèmes critiques nous a permis de constater**
 - ✓ qu'une liste de méthodes communes peuvent être appliquées pour adapter et intégrer le LL dans le système critique.
 - ✓ Ces méthodes dépendent de la criticité du logiciel et peu du secteur industriel concerné.
- **Des solutions d'architecture dédiées**
 - ✓ Encapsulation
 - ✓ Partitionnement
 - ✓ Mécanismes de protection : Sécurité-confidentialité
- **Architecture basée sur des standards d'interface**
 - ✓ Favorise les LL
 - ✓ Apporte une bonne durée de vie pour la maintenance

Gains espérés de l'utilisation des LL: le cercle vertueux

- L'utilisation de standards d'interface permettra **d'échanger des logiciels**
- L'effort de certification, qui demande un gros investissement initial pour la constitution du dossier, **peut être partagé.**
- La frilosité des industriels vient de la non compatibilité apparente des LL avec le processus de certification
 - ✓ Nous avons démontré l'inverse avec le support des solutions architecturales
 - ✓ Les risques sont mieux gérés que pour les COTS
 - ✓ Le dossier de certification doit être initialisé par un groupe d'industriels
 - ✦ mise en commun du dossier de certification
 - ✦ Ce dossier pourra être mis à disposition de façon à s'enrichir des expériences d'autres industriels.
- **Amorçage du cercle vertueux**



Conclusion

- l'utilisation de COTS présente des limitations
- l'emploi de LL apporte de vrais avantages:
 - ✓ respect des standards
 - ✓ moindre risque
 - ✓ code source pour la maintenance
- **Le principal inconvénient**
 - ✓ incompatibilité apparente avec le processus de certification
- ➔ **solutions architecturales**
- ➔ **atelier d'adaptation des LL aux méthodes industriels pour mise à niveau**
- ➔ **Amorçage du cercle vertueux pour une initiative industrielle**
- ➔ **Contribution à la communauté du libre**

Exemple d'architectures

- **Spatial**
- **Aéronautique**

Architecture en niveaux (1)

La mise en œuvre de composant sur étagère (COTS ou LL) s'accompagne généralement d'une architecture logicielle à 3 niveaux

- **Niveau 1** : environnement d'exécution et d'abstraction des ressources matérielles et logicielles,
- **Niveau 2** : services standardisés ou propriétaires permettant l'exécution des applications,
- **Niveau 3** : Structure d'accueil de haut niveau pour les applications

Architecture en niveaux (2)

- **Utilisation de COTS ou LL en majorité au niveau 1**
 - ✓ exécutif temps réel, pile de communication bibliothèque d'interface graphique, ...)
 - ✓ l'atelier de production et de validation semble être un domaine où le logiciel libre fait une percée importante
- **Le niveau 2 est une réponse architecturale à l'intégration des COTS ou LL pour en maîtriser l'utilisation, faute de pouvoir en maîtriser la conception.**
 - ✓ Permet la mise en œuvre de mécanismes de tolérance aux fautes et de confinement d'erreur
 - ✦ *Wrapper*
 - ✦ Définition des partitions
 - ✓ Permet de limiter la dépendance des applications vis à vis du matériel et donc de les pérenniser
- **Le niveau 3 dépend du domaine d'application**
 - ✓ chaque industriel dispose de solutions adaptées à son contexte.

Le domaine spatial:

Utilisation de RTEMS

Contexte

- **Nouvelle génération de calculateurs « 32 bits »**
 - ✓ Développements en langage C
 - ✓ Nécessité d'identifier un « Exécutif temps-réel » compatible
- **Caractéristiques du système spatial (bord)**
 - ✓ Difficilement maintenable, son efficacité opérationnelle doit être garantie
 - ✓ Néanmoins soumis (de plus en plus) aux contraintes économiques et calendaires
 - ✓ Tendance à la standardisation (interfaces et services)
- **Trois solutions**
 - ✓ Produit sur étagère
 - ✦ Expérience existante sur vxWorks: Columbus, Proba
 - ✓ Logiciel Libre
 - ✦ Expérience existante sur RTEMS
 - ✦ Utilisation de RTEMS encouragée par ESA
 - ✓ Développement d'une solution spécifique

Pourquoi RTEMS ?

➤ Solution spécifique écartée

- ✓ Génération de processeurs « 32bits » complexe (?)
- ✓ Expertise requise non rentabilisable dans le domaine spatial

➤ Logiciel libre préféré

- ✓ Contexte international
- ✓ Souplesse dans la configuration d'un produit répondant au juste besoin (restriction des services)
- ✓ Souplesse dans l'élaboration et la mise en place d'une politique de maintenance, de pérennisation, et de déploiement
 - ✦ Source disponible, droits de modification & livraison
 - ✦ Pas d'engagement sur l'image d'un fournisseur
- ✓ Normalisation à l'échelle européenne pouvant permettre d'améliorer la qualité des produits

Maintenance

- **L'industrie se propose de maintenir le produit**
- **Mise en place d'un processus « produit RTEMS »**
 - ✓ Plate-forme d'exécution de référence (autour du processeur ERC32SC)
 - ✓ Gestion en configuration indépendamment des projets utilisateurs
 - ✓ Suivi de la version « publique »
 - ✓ Gestion des Demandes de Modification issues des projets utilisateurs
- **Maintient de la compétence par les projets utilisateurs**
- **Effort de documentation du produit en conformité avec les règles standard des clients**
 - ✓ Création des Plan de développement et Spécification logicielle
 - ✓ Création du Plan de validation et mise en place d'une traçabilité
 - ✓ Complément du Manuel utilisateur existant
 - ✓ Pas de document de conception
- **Environnement de développement**
 - ✓ Utilisation des outils GNU disponibles avec RTEMS
 - ✓ Utilisation de moyens spécifiques pour les test temps-réel

**Le candidat Linux pour l'avionique
embarquée ?**

Contexte avionique embarquée

- **Questions posées : Linux peut-il ...**
 - ✓ Être embarqué dans du matériel avionique non critique
 - ✓ Être adapté pour répondre aux propriétés avioniques
 - ✓ Entrer dans un processus de certification DO178B
- **Contraintes imposées au niveau du programme avion**
 - ✓ Banalisation ressources de calculs (moins d'éléments de rechange)
 - ✓ Réseau avion basé sur AFDX (Ethernet commuté déterministe)
- **Propriétés requises**
 - ✓ Partitionnement mémoire
 - ✓ Partitionnement temporel
 - ✓ Partitionnement E/S
- **A653 => perte d'interopérabilité, nécessite développement d'outils spécifiques (réutilisation difficile)**
- ➔ **Utiliser Linux comme plate-forme d'accueil d'applications avioniques assurant l'interopérabilité et garantissant les contraintes de partitionnement**

Bilan des études

➤ Points forts de Linux

- ✦ Bonne lisibilité du code du noyau, bonne expertise (support de la communauté)
- ✦ Points communs avec LynxOS
- ✦ Interopérabilité (standards industriels)
- ✦ Environnement d'outils GNU complet pour le développement

➤ Du point de vue de la certification ...

- ✦ Pas de points techniques bloquants vis-à-vis des contraintes de l'avionique embarquée
- ✦ Nécessité d'un contrôle d'allocation des ressources système et du remplacement des mécanismes non « avionables »
- ✦ Mise en conformité du code vis-à-vis des standards de codage avioniques, peu de document de conception du système, documentation variée, mais incomplète, éparse, et pas forcément à jour
- ✦ Coût d'appropriation non négligeable, nécessite un effort de « reverse engineering » pour recréer le matériel de certification

Problématique

- **Connaissance partielle des caractéristiques du logiciel libre**
 - ✓ Hypothèses de conception
 - ✓ Performances
 - ✓ Propriétés de Sûreté de Fonctionnement : **modes de défaillances**
 - ✓ Fonctions cachées
 - ✓ Interopérabilité...
- **Evolutions fréquentes, pas toujours contrôlées**

Nécessité de principes architecturaux pour les intégrer dans des systèmes critiques

Problématique est similaire à celle d'intégration des COTS

Proposition pour l'utilisation de LL
dans les systèmes critiques

Solutions d'architecture

- L'utilisation des LL est prometteuse et est une tendance lourde
- Les techniques doivent être consolidées pour les introduire dans nos systèmes
 - ✓ L'encapsulation: isoler le LL pour mieux gérer ses versions
 - ✓ L'empaquetage: maîtriser les modes de défaillances du LL
 - ✓ Le partitionnement des fonctions
 - ✓ Les mécanismes de protection: sécurité-confidentialité
- Des solutions complètes d'architecture doivent être définies et qualifiées (famille par niveau de criticité)
 - ✓ Exemple d'une IMA ouverte, en réseau avec d'autres systèmes, redondées, LL (linux, protocoles) au niveau C ou D.

Un processus de développement à formaliser

- **La certification commande des actions**
 - ✓ De développement
 - ✓ D'évaluation
 - ✓ De documentation
 - ✓ De test
- **Il faut définir un atelier de mise à niveau des LL qui soit transverse aux domaines industriels**
 - ✓ Identifier des méthodes communes
 - ✓ Amorcer la dynamique de partage
 - ✓ Mettre les outils correspondants à disposition des volontaires sur un site
- **Initier une structure d'évaluation et de capitalisation des LL**
 - ✓ Validation : caractérisation des modes de défaillances
 - ✓ Mise a disposition des empaquetages
 - ✓ Configuration des LL
 - ✓ Initier la mise en place de dossier de certification en commun

Un processus futur à préparer

- Le logiciel n'est pas le seul support du libre
- LL signifie aussi Hardware Libre (VHDL ou modèles C)
- L'ingénierie système évolue vers la notion de modèle et non plus de logiciel comme l'entité exécutable
- Il faut transcrire cela dans le monde du libre:
 - ✓ Atelier de production système libre
 - ✓ Création de modèles libres

Une nouvelle licence

- Les besoins industriels ou des administrations peuvent être en désaccord avec la GPL.
- Il faut travailler l'idée d'une licence qui facilite l'utilisation de LL dans le milieu industriel
- Initiative au niveau Européen