

**L4.3: Compte-rendu Essais de Certification Environnement
- Outils de Développement**



Document rédigé par :	Approuvé par :	Approuvé par :	Approuvé par :
Claude Guilmain (Sagemcom) Olivier Gallot (Axupteam)			

**Compte-rendu Essais de Certification Environnement –
Outils de Développement
Liste des évolutions**

Edition	Date	Evolutions
Edition 01	16 Mai 2011	Création

RTEL4I_Outils_Environnement

Fiche de rapport de test

Projet: RTEL4I_Environnement_Outils
Auteur : Claude GUILMAIN
Imprimé par TestLink le 17/05/2011
2009 © TestLink Community

Table des matières

Plate-forme:SODIMM2410

Analyse

Analyse d'événements

Analyse LTT

Analyse mauvaise utilisation mémoire

Analyse Oprofile

Analyse licences

Debug

Debug d'une application dans le domaine utilisateur

Debug du noyau

Debug par JTAG

Debug avec Kprobes

Exploitation de crash dump noyau

Debug avec simulateur Xenomai

Edition

Choix du temps réel

Ajout d'un module

Codes sources avec syntaxe colorée

Navigation dans le code source

Gestion de Configuration

Gestion du plan de version

Gestion des recettes

Gestion des sources

Génération

Plan de version d'un firmware

Profils du firmware

Génération unitaire d'un outil

Re-génération suite à modification

Génération unitaire d'un module

Composition variable du firmware

Option spécifique à un profil

Re-génération unitaire d'un module

Re-génération du firmware

Génération parallélisée

Recette

module avec option spécifique à un profil

Option générique pour tous les profils

Nettoyage complet du firmware

Nettoyage d'un module du firmware

Nettoyage du système de fichiers cible du firmware

Mesure

- mesure de la mémoire
- mesure de latence et gigue
- mesure de la charge du système
- mesure de débit

Synthèse des résultats

1.1. Test Suite : Analyse

Cette suite de tests traite de l'analyse des logiciels qui composent le Firmware

Cas de Tests RTEL4I_EO-26: Analyse d'événements		
<u>Auteur :</u>		Olivier GALLOT
<u>Résumé:</u> Tracer le flot d'événements qui s'exercent au sein du système lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur met en œuvre l'outil Ftrace: configuration noyau CONFIG_FUNCTION_TRACER exécute le firmware recueille les événements traçables cat /sys/kernel/debug/tracing/available_events trace tous les événements disponibles echo *.* > /sys/kernel/debug/tracing/set_event	L'utilisateur trace l'ensemble des événements qui s'exercent dans le système: interruptions, réveil de tache, changement de tache, signaux, allocation mémoire ...
2	L'utilisateur met en œuvre l'outil SystemTap: ajout de systemtap dans le générateur de firmware configuration noyau CONFIG_DEBUG_INFO, CONFIG_KPROBES, CONFIG_RELAY, CONFIG_DEBUG_FS, CONFIG_MODULES, CONFIG_MODULE_UNLOAD, CONFIG_UTRACE	L'utilisateur trace les événements synchrones et asynchrones.
3	L'utilisateur met en œuvre l'outil perf exploitant	L'utilisateur trace les

Compte-rendu Essais de Certification Environnement – Outils de Développement

	<p>le sous système noyau "Performance Counter": ajout de l'outil perf dans le générateur pour l'embarquer dans le firmware cible configuration du noyau CONFIG_PERF_COUNTER=y, CONFIG_EVENT_TRACING=y L'utilisateur liste les événements disponibles: perf list L'utilisateur exécute un enregistrement d'événements choisis: perf record</p>	<p>interruptions, les exceptions, les signaux, les allocations mémoire, les changements de taches, ...</p>
<u>Dernier résultat:</u>	Non Disponible	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Outils non disponibles pour l'architecture ARM.	
<u>Exigences fonctionnelles</u>	EX_ 48: Avoir une vue sur les évènements internes de la machine	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-27: Analyse LTT		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Analyser les événements au sein du système avec LTT (Linux Trace Toolkit) lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur met en œuvre Linux Traces Toolkit: applique le patch ltt-ng correspondant à sa version de noyau Linux</p> <p>configure le noyau: Activate Markers CONFIG_DEBUG_INFO,CONFIG_RELAY, CONFIG_DEBUG_FS</p> <p>compilation des modules ltt-trace-control, ltt-tracer</p> <p>génération complète du firmware</p> <p>L'utilisateur exécute le firmware, configure le contrôleur pour tracer les événements de son choix, exécute l'application de traitement de flux sur une période choisie à l'issue de laquelle il stoppe et extrait les traces pour finir par les visualiser dans lttv.</p>	<p>L'utilisateur visualise les événements du système: interruptions, appel systèmes, ordonnancement, utilisation de la pile réseau ...</p>
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le patch et le package LTT pour la version noyau 2.6.25 ne sont pas intégrés dans le générateur buildroot.	
<u>Exigences fonctionnelles</u>	EX_ 49: Avoir une vue sur l'enchaînement des tâches EX_ 50: Avoir une vue sur les appels systèmes EX_ 51: Avoir des chronogrammes avec changement d'échelle EX_ 51.1: Avoir une colorisation des événements	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-28: Analyse mauvaise utilisation mémoire		
<u>Auteur :</u>		Olivier GALLOT
<u>Résumé:</u> Détecer les mauvaises utilisation de la mémoire dans le système embarqué.		
<u>Preconditions:</u> L'application déboguée peut être celle de traitement de flux de la suite de tests temps réel albatros.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	Mise en œuvre de mudflap : l'utilisateur compile et lie l'application à déboguer avec les options -fmudflap et -lmudflap.	Suivant la configuration mudflap choisie: l'application fautive génère un signal SIGSEGV l'application fautive appelle abort() et génère un coredump l'application fautive ouvre une session gdb sur le programme suspendu.
2	Mise en œuvre de duma : l'utilisateur compile et lie l'application avec la librairie libduma. Il l'exécute enfin dans une session de debug GDB	Lors d'une faute mémoire de l'application déboguée (accès par erreur en écriture ou lecture à une page mémoire allouée par duma) le signal SIGSEGV est levé puis attrapé par gdb. Depuis le débogueur, l'utilisateur peut examiner le code ayant abouti à l'erreur pointée.
3	mise en œuvre de mtrace : l'utilisateur ajoute dans le code main() de l'application à déboguer la référence à mtrace(), mcheck.h puis compile avec l'option -g.	L'utilisateur dispose d'une trace des blocs de mémoire alloués dynamiquement non libérés par l'application déboguée.
4	mise en oeuvre de kmemleak : détecteur de fuite mémoire au sein du code noyau. L'utilisateur recompile le noyau avec l'option: Kernel Hacking -> Kernel memory leak detector. L'utilisateur monte le système de fichier debugfs. # mount -t debugfs nodev /sys/kernel/debug	L'utilisateur accède facilement aux fuites mémoire du code noyau: # cat /sys/kernel/debug/memleak

<u>Dernier résultat:</u>	Echec
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1
<u>Tester</u>	Olivier GALLOT
<u>Notes d'exécution</u>	Les modules duma, mtrace, mudflaps ne sont pas intégrés et documentés dans le générateur buildroot. Le module valgrind intégré dans le générateur ne fonctionne pas sur l'architecture ARM.
<u>Exigences fonctionnelles</u>	EX_ 52: Pouvoir detecter les mauvaises utilisations de memoire
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-29: Analyse Oprofile		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Localiser les codes pour lesquels le processeur consomme le plus de temps.	
<u>Preconditions:</u>	L'application analysée peut être celle de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>Mise en oeuvre de Oprofile: l'utilisateur compile le noyau avec "Oprofile system profiling" et kernel debugging "Debug info".</p> <p>Il embarque les outils opcontrol et oprofile dans le firmware cible, configure et démarre leur enregistrement avant d'exécuter l'application à analyser:</p> <pre>opcontrol --init opcontrol --setup --separate=all --vmlinux=/root/vmlinux opcontrol --start</pre> <p>L'utilisateur récupère les traces qui l'intéresse:</p> <pre>oprofile --merge=all -l</pre>	L'utilisateur visualise la répartition du temps CPU dans les différentes composantes logicielles de son application.
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Avec le générateur buildroot l'utilisateur ne peut pas embarquer dans le firmware les binaires opcontrol et oprofile s'il utilise une toolchain externe.	
<u>Exigences fonctionnelles</u>	EX_ 53: Pouvoir faire du profiling	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-34: Analyse licences		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Présenter les licences des modules du firmware.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur demande la collecte des licences des codes sources des binaires embarqués dans le firmware cible avant de lancer la génération de celui-ci.	L'utilisateur dispose de l'ensemble des licences utilisées par le firmware construit.
<u>Dernier résultat:</u> Echec		
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	Aucun outil intégré à buildroot permet de recueillir les licences des binaires embarqués dans le firmware généré.	
<u>Exigences fonctionnelles</u>	EX_ 58: Il doit etre possible de collecter des informations sur les modules et outils references par le pla	
<u>mots clés:</u>	Aucun	

1.2. Test Suite : Debug

Cette suite de tests traite du Debug d'applications, de code noyau et des outils associés

Cas de Tests RTEL4I_EO-20: Debug d'une application dans le domaine utilisateur		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Une application embarquée sur la cible est exécutée dans le debugger gdb.		
<u>Preconditions:</u> La cible peut être émulée dans Qemu. L'application multi-threads déboguées peut être celle de traitement de flux de la suite de tests temps réel albatros.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>Session gdb / gdbserver Le programme gdbserver est embarqué et exécuté sur le système de fichiers cible. L'utilisateur s'y connecte depuis son pc avec le client gdb cross compilé pour la plateforme cible.</p> <p>Session gdb natif Le programme gdb compilé pour la plateforme cible est embarqué sur le système de fichiers cible. L'utilisateur l'exécute sur la cible qui dispose de suffisamment de mémoire vive pour garantir le bon déroulement de la session de debug.</p>	<p>L'utilisateur peut déboguer des applications multi-threads: tous les threads créés par l'application sont visibles et débogables.</p> <p>L'utilisateur peut déboguer des applications qui réalise des appels systèmes fork() ou vfork(): il peut suivre au choix le processus fils ou père.</p> <p>L'utilisateur peut déboguer comme ci-dessus une application exécuté dans le système émulé par Qemu. Sans changer les caractéristiques énoncées ci dessus, le système de fichier cible peut être local à la plateforme ou distant et accédé par le réseau.</p>
<u>Dernier résultat:</u> Reussi		
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	<p>L'utilisateur embarque gdbserver sur la cible et l'exécute sur le programme albatros-01/network/rt_receive.</p> <p>Il se connecte à gdbserver avec le client gdb de la toolchain externe et commande l'exécution du test rtp_receive.</p> <p>Il exécute albatros/network/rtp_rend sur une machine distante en direction de la cible.</p> <p>L'application rtp_receive trap sur "Illegal instruction" clz et l'utilisateur reprend la main dans gdbserver.</p>	

Compte-rendu Essais de Certification Environnement – Outils de Développement

<u>Exigences fonctionnelles</u>	EX_ 45: Pouvoir debugger des applications EX_ 45.1: directement sur la cible EX_ 45.2: sur une cible emulee
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-21: Debug du noyau		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Debugger le code noyau avec Kgdb.	
<u>Preconditions:</u>	<p>La cible peut être émulée dans Qemu.</p> <p>La session de debug du noyau peut être réalisé lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.</p>	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur configure le noyau pour être débogué depuis un client gdb distant:</p> <p>Kernel hacking ---> KGDB: kernel debugging with remote gdb</p> <p>L'utilisateur choisit suivant son besoin la méthode de communication de kgdb: lien serie ou ethernet.</p>	<p>L'utilisateur debogue le code noyau au travers du lien de communication choisi (serie ou ethernet): insertion de points d'arrêt, affichage des variables, des registres et de la pile d'exécution jusqu'au "program counter" courant.</p> <p>L'utilisateur débogue le code noyau du système émulé dans Qemu sans restriction.</p>
<u>Dernier résultat:</u>	Echec	
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	Le patch kgdb n'est pas intégré au générateur buildroot.	
<u>Exigences fonctionnelles</u>	<p>EX_ 46: Pouvoir debugger au niveau du kernel</p> <p>EX_ 46.1: directement sur la cible.</p> <p>EX_ 46.2: sur une cible emulee.</p>	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-22: Debug par JTAG		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Debugger le code noyau par JTAG.	
<u>Preconditions:</u>	<p>Une sonde JTAG est disponible pour la cible.</p> <p>La session de debug noyau peut être réalisée lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.</p>	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur dispose d'un adaptateur USB vers JTAG type FTDI FT2232: il installe openOCD sur son poste de développement.</p> <p>A l'identique d'une session gdb/gdbserver, l'utilisateur exécute le démon openocd, ouvre un session gdb pour le binaire vmlinux et se connecte au port écouté par openocd.</p> <p>L'utilistateur place un point d'arrêt dans le code et examine le contenu des registres du processeur cible.</p>	<p>Debugger par JTAG afin de préserver le système de l'utilisation des interfaces génériques usb, ethernet ...</p>
<u>Dernier résultat:</u>	Non Disponible	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Sonde JTAG non disponible	
<u>Exigences fonctionnelles</u>	<p>EX_ 46: Pouvoir debugger au niveau du kernel</p> <p>EX_ 46.3: sur la cible via jtag</p>	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-23: Debug avec Kprobes		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Insertion de probes dans le noyau et recueil des traces.	
<u>Preconditions:</u>	Les traces peuvent être recueillies lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur instrumente le code noyau et celui des modules pour insérer selon le besoin:kprobes,jprobes ou kretprobes. Il compile le noyau avec les options suivantes: CONFIG_KPROBES,CONFIG_MODULES,CONFIG_MODULE_UNLOAD, CONFIG_KALLSYMS_ALL pour produire le firmware cible.	L'utilisateur trace l'exécution du noyau avec les "probes".
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Developpement Version 1	
Tester	Olivier GALLOT p778206	
Notes d'exécution	<p>L'utilisateur active les Kernel Probes CONFIG_KPROBES=y, CONFIG_KPROBES_SANITY_TEST=y dans le fichier de configuration du noyau board/pragmatec/sodimm2410/config_SODIMM2410_OW_2.6.25_xeno puis lance la génération du firmware.</p> <p>L'exécution du firmware échoue dans la phase de démarrage avec l'erreur suivante:</p> <pre> Kprobe smoke test started Unable to handle kernel paging request at virtual address cf4984ec pgd = c0004000 [cf4984ec] *pgd=00000000 Internal error: Oops: 5 [#1] Modules linked in: CPU: 0 Not tainted (2.6.25-pragmatec-adeos-dirty #1) PC is at __ipipe_dispatch_event+0xdc/0x20c LR is at __und_svc+0x2c/0x60 pc : [] lr : [] psr: a0000093 sp : c3c1be80 ip : 00000000 fp : c3c1bedc r10: c03cdec8 r9 : c3c1bf68 r8 : c0428740 </pre>	

Compte-rendu Essais de Certification Environnement – Outils de Développement

	<pre> r7 : c03cdec8 r6 : c0428744 r5 : c03cdeb8 r4 : c0428740 r3 : 0f06fda8 r2 : 60000093 r1 : 0f06fda8 r0 : ffffffff Flags: NzCv IRQs off FIQs on Mode SVC_32 ISA ARM Segment kernel Control: c000717f Table: 30004000 DAC: 00000017 Process swapper (pid: 1, stack limit = 0xc3c1a268) Stack: (0xc3c1be80 to 0xc3c1c000) be80: c03cdec8 c3c1bf68 6f72706b 745f6562 ffffffff ffffffff 00746e6d c3c1bee0 bea0: c0428740 0f06fda8 00000000 00000000 c3c10070 ffffffff c3c1bf14 c3c1a000 bec0: 00000000 00000000 00000000 00000000 c3c1bf7c c3c1bee0 c02fdb8c c0078028 bee0: 9f82bf91 00000000 00000001 00000000 c0428258 c0428258 c3c1a000 00000000 bf00: 00000000 00000000 00000000 c3c1bf7c c3c1bf30 c3c1bf68 c006df2c c006de48 bf20: 60000013 ffffffff c03008c0 c02feb24 c02fd780 c006df1c c0428258 c0428258 bf40: c3c1a000 00000000 c3c1bf64 c3c1bf58 c0300ae4 c03005a4 c3c1bf7c c3c1bf68 bf60: c006df1c c0300ae0 00000000 c0021c3c c3c1bf94 c3c1bf80 c0013684 c006dec bf80: 00000000 00000000 c3c1bff4 c3c1bf98 c00085f0 c001363c 00000000 00000000 bfa0: 00000000 c3c1bfb0 00000001 c0041c80 00000000 00000000 c0008514 c0049734 bfc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 bfe0: 00000000 00000000 00000000 c3c1bff8 c0049734 c0008524 00800000 01000000 Backtrace: [] (__ipipe_dispatch_event+0x0/0x20c) from [] (__und_svc+0x2c/0x60) [] (init_test_probes+0x0/0x220) from [] (init_kprobes+0x58/0x6c) r5:c0021c3c r4:00000000 [] (init_kprobes+0x0/0x6c) from [] (kernel_init+0xdc/0x2b4) r4:00000000 r3:00000000 [] (kernel_init+0x0/0x2b4) from [] (do_exit+0x0/0x7d0) Code: e1a06007 e1a07003 e51b1038 e2464004 (e7963001) ---[end trace 778e504de7e3b1e3]--- Kernel panic - not syncing: Attempted to kill init! </pre>
<u>Exigences fonctionnelles</u>	EX_ 46: Pouvoir debugger au niveau du kernel
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-24: Exploitation de crash dump noyau		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Génération et exploitation d'un crash dump noyau.		
<u>Preconditions:</u> Le crash noyau peut être expérimenté lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur crée pour la plateforme cible un système de fichiers distant accédé par le réseau. Ce système de fichier embarque les outils kexec/kdump. Le firmware fait le montage nfs d'un répertoire du pc de développement pour recevoir le crash dump du noyau.	Les traces du noyau en erreur sont enregistrées dans le répertoire désigné et le système a redémarré sur le noyau kdump de secours.
<u>Dernier résultat:</u> Non Disponible		
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Les outils Kexec/Kdump ne sont pas disponibles pour l'architecture ARM et la version noyau 2.6.25.	
<u>Exigences fonctionnelles</u>	EX_ 46: Pouvoir debugger au niveau du kernel	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-25: Debug avec simulateur Xenomai		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Insertion d'une application dans le simulateur Xenomai.	
<u>Preconditions:</u>	L'application peut être celle de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur charge une application temps réel xenomai dans le simulateur.	L'utilisateur exécute pas à pas le code dans le simulateur
<u>Dernier résultat:</u>	Non Disponible	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le patch Xenoscope n'est pas implémenté pour l'architecture ARM.	
<u>Exigences fonctionnelles</u>	EX_ 47: Pouvoir mettre en Suvre facilement le simulateur de Xenomai EX_ 47.1: Pouvoir inserer facilement des applications dans le simulateur	
<u>mots clés:</u>	Aucun	

1.3. Test Suite : Edition

Cette suite de tests traite des aides apportées par l'IDE dans l'édition des fichiers sources

Cas de Tests RTEL4I_EO-4: Choix du temps réel		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Le champs RTOS du plan de version du firmware désigne le type de temps réel mis en oeuvre.		
<u>Preconditions:</u> L'utilisateur a le choix parmi plusieurs solutions temps réel et plusieurs skins Xenomai. Les valeurs possibles du champs RTOS sont les suivantes: PREEMPT-RT , Xenomai , NULL Les valeurs possibles du champs Xenomai Skin sont les suivantes: Posix , VxWorks , vrtx , psos Les modules du firmware utilisant les skins posix et vxworks peuvent être ceux de l'application de traitement de flux de la suite de tests temps réel albatros.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur édite le plan de version du firmware, affecte le champs RTOS à PREEMPT_RT, puis lance la génération du firmware cible.	Le générateur applique le patch PREEMPT_RT au code source du noyau et construit le firmware temps réel mou composé des modules décrits par le plan de version.
2	L'utilisateur édite le plan de version du firmware, affecte le champs RTOS à Xenomai, puis lance la génération du firmware cible.	Le générateur applique le patch Xenomai au code source du noyau et construit le firmware temps réel dur composé des modules décrits par le plan de version.
3	L'utilisateur affecte le champs RTOS du plan de version à Xenomai et le champs Xenomai Skin d'un module du firmware à Posix.	Le générateur applique le patch Xenomai au code source du noyau et compile le module designé ci-contre suivant la skin posix.
4	L'utilisateur réitère le point précédent avec la skin vxworks.	Le générateur construit le firmware temps réel dur xenomai qui embarque un module codé suivant l'API vxworks.
<u>Dernier</u>	Echec	

<u>résultat:</u>	
Session	Essais de Certification Environnement et Outils de Développement Version 1
Tester	Olivier GALLOT
Notes d'exécution	<p>L'utilisateur choisi le plan de version sodimm2410_glibc_2625_xeno_sourcery_defconfig pour construire le firmware temps réel xenomai.</p> <p>Aucun plan de version preempt_rt de disponible, l'utilisateur suit les étapes suivantes pour le mettre en oeuvre:</p> <ul style="list-style-type: none"> + définition d'un fichier de configuration du noyau config_SODIMM2410_OW_2.6.25_EABI_preempt_rt avec les parametres suivants: <pre>CONFIG_PREEMPT_RT=y CONFIG_PREEMPT=y CONFIG_PREEMPT_SOFTIRQS=y CONFIG_PREEMPT_HARDIRQS=y</pre> <ul style="list-style-type: none"> + référence du package PREEMPT_RT et du fichier config_SODIMM2410_OW_2.6.25_EABI_preempt_rt dans le fichier plan de version (.config): <pre>BR2_PACKAGE_PREEMPT_RT=y BR2_PREEMPT_RT_KCONFIG="board/pragmatec/sodimm2410/config_SODIMM2410_OW_2.6.25_EABI_preempt_rt" BR2_PREEMPT_RT_VERSION=".4-rt1"</pre> <ul style="list-style-type: none"> + construction du firmware <p>Quelque soit la valeur de BR2_PREEMPT_RT_VERSION: .4-rt1 à .4-rt6 ou .8-rt7 l'application du patch preempt_rt échoue.</p> <p>Aucune version du patch preempt-rt spécifique à la carte sodimm2410 est disponible dans le générateur buildroot fourni.</p> <p>En conclusion: le générateur de firmware facilite guère la mise en oeuvre des solutions temps réel concurrentes Xenomai et preempt_rt.</p> <p>Avec la version buildroot livrée la mise en oeuvre de PREEMPT_RT échoue.</p>
<u>Exigences fonctionnelles</u>	<p>EX_ 2: Pouvoir facilement mettre en Suvre Xenomai ou Premp-RT</p> <p>EX_ 23: Pouvoir mettre en Suvre facilement une distribution temps reel</p> <p>EX_ 23.1: Pouvoir modifier facilement le plan de version pour avoir Xenomai</p> <p>EX_ 23.2: Pouvoir modifier facilement le plan de version pour choisir les skins de Xenomai</p> <p>EX_ 23.3: Pouvoir modifier facilement le plan de version pour avoir Premp-RT</p>
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-6: Ajout d'un module		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> L'utilisateur édite le plan de version pour ajouter un module dans la composition du firmware.		
<u>Preconditions:</u> L'utilisateur dispose d'un serveur de gestion de configuration (svn,git,mercurial). Chaque firmware dispose d'un répertoire contenant son plan de version et le Makefile pour lancer sa construction. Le module ajouté peut être la suite de tests temps réel albatros.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le plan de version d'un firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA/plan_version.txt	L'utilisateur dispose d'une copie locale editable du plan de version du firmware.
2	Depuis un shell unix et d'une seule commande, l'utilisateur lance la construction du firmware référencé par le plan de version.	L'utilisateur dispose du binaire de référence du firmware désigné par le plan de version archivé dans la gestion de configuration.
3	L'utilisateur ouvre le fichier plan de version du firmware dans son éditeur favori ou dans l'IDE eclipse, ajoute la référence à son nouveau module et reconstruit le firmware.	L'utilisateur dispose d'un nouveau binaire firmware qui embarque le module ajouté comme désigné dans la nouvelle version du plan de version du firmware. Lors de cette nouvelle génération du firmware, seules les codes sources du module ajouté et de ses dépendances sont compilés.
4	L'utilisateur édite à nouveau le plan de version du firmware, retire la référence au module ajouté ci-dessus puis relance la construction du firmware avec la seule et même commande qu'au début du test.	L'utilisateur dispose à nouveau du binaire de référence du firmware désigné par le plan de version archivé dans la gestion de configuration. Les binaires obtenus à ce point et lors de la première génération diffèrent uniquement par le contenu des chaînes de caractères qui enregistrent la date de construction des binaires.
<u>Dernier</u>	Reussi	

<u>résultat:</u>	
Session	Essais de Certification Environnement et Outils de Développement Version 1
Tester	Olivier GALLOT
Notes d'exécution	<p>L'utilisateur ajoute le module albatros au générateur.</p> <p>Il crée le répertoire package/albatros et enregistre la recette albatros.mk dans laquelle il référence l'URL pour récupérer les sources albatros-1.0.tar.gz et la dépendance au package xenomai.</p> <p>Il crée le fichier package/albatros/Config.in qui documente et définit la référence du module albatros dans le générateur BR2_PACKAGE_ALBATROS.</p> <p>Enfin il insère albatros parmi les packages du générateur: package/Config.in:source "package/albatros/Config.in"</p> <p>L'utilisateur peut le construire "make albatros" et/ou l'insérer dans le plan de version BR2_PACKAGE_ALBATROS=y d'un firmware temps réel xenomai.</p>
<u>Exigences fonctionnelles</u>	<p>EX_ 11: Le plan de version doit pouvoir être lu et généré par l'IDE.</p> <p>EX_ 12: Le plan de version doit pouvoir être exploité par une commande en ligne depuis un simple shell</p> <p>EX_ 13: Le plan de version doit être lisible par un humain et éditable</p> <p>EX_ 14: Le plan de version doit pouvoir être géré dans une gestion de configuration</p> <p>EX_ 15: Le format du plan de version est indépendant de tout format eclipse</p> <p>EX_ 21: Pouvoir modifier de façon dynamique les divers éléments du plan de version</p> <p>EX_ 21.2: Pouvoir ajouter, retirer, modifier un des composants logiciels du firmware</p> <p>EX_ 21.3: Ne pas avoir à tout systématiquement re-générer lorsque le plan de version est modifié</p>
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-37: codes sources avec syntaxe colorée		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Les codes sources chargées dans l'éditeur de l'IDE sont colorées suivant le langage.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur ouvre dans l'IDE un fichier Makefile d'un module du firmware.	L'éditeur de l'IDE présente les cibles et règles codées dans le fichier Makefile suivant un code couleur.
2	L'utilisateur ouvre dans l'IDE un fichier c++ d'un module du firmware.	L'éditeur de l'IDE présente les types de donnés et les mots clés du langage c++ avec des couleurs différentes.
3	L'utilisateur ouvre dans l'IDE un fichier python d'un module du firmware.	L'éditeur de l'IDE présente les types de donnés et les mots clés du langage python avec des couleurs différentes.
<u>Dernier résultat:</u>		Non Disponible
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le module Eclipse du projet RTEL4I n'est pas livré.	
<u>Exigences fonctionnelles</u>	EX_ 27: Avoir la possibilite d'editer les sources dans les meilleures conditions possibles EX_ 27.1: colorisation syntaxique en fonction du langage	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-38: Navigation dans le code source		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Depuis l'IDE, parcourir facilement la structure d'un programme dont les codes sources sont répartis dans plusieurs fichiers.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur ouvre dans l'IDE un fichier source codé en c++ d'un module du firmware sélectionné. Il parcourt le code d'une fonction implémentée dans le fichier et clique avec sa souris sur le type de donnée d'une variable utilisée dans la fonction. Le fichier qui définit le type de donnée est ouvert dans l'IDE.	L'utilisateur parcourt facilement (avec sa souris ou une combinaison de touches de son clavier) la structure des programmes.
<u>Dernier résultat:</u>	Non Disponible	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le module Eclipse du projet RTEL4I n'est pas livré.	
<u>Exigences fonctionnelles</u>	EX_ 27: Avoir la possibilité d'éditer les sources dans les meilleures conditions possibles EX_ 27.3: Naviguer facilement dans le code d'un projet	
<u>mots clés:</u>	Aucun	

1.4. Test Suite : Gestion de Configuration

Cette suite de tests couvre les besoins de gestion de configuration

Cas de Tests RTEL4I_EO-10: Gestion du plan de version		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Le plan de version du firmware est enregistré dans un outil de gestion de configuration.	
<u>Preconditions:</u>	L'utilisateur dispose d'un serveur de gestion de configuration (svn,git,mercurial). Chaque firmware dispose d'un répertoire contenant son plan de version et le Makefile pour lancer sa construction.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur extrait le firmware d'un système de gestion de configuration.</p> <pre>svn co http://svn.sagemcom.com/products/firmwareA</pre> <p>Il édite le fichier plan_version.txt , change le nom du firmware et enregistre cette nouvelle version du plan de version du firmware dans le système de gestion de configuration</p> <pre>svn ci http://svn.sagemcom.com/products/firmwareA/plan_version.txt -m "change name to firmwareA+"</pre>	L'utilisateur peut gérer les différentes versions du plan de version du firmware dans un système de gestion de configuration.
<u>Dernier résultat:</u>	Reussi	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	<p>L'utilisateur édite le plan de version sodimm2410_glibc_2625_xeno_sourcery_defconfig et ajoute le package procps pour embarquer l'utilitaire pmap BR2_PACKAGE_PROCPS=y.</p> <p>Il enregistre dans la gestion de configuration git cette nouvelle version du plan de version.</p>	
<u>Exigences fonctionnelles</u>	EX_ 28: Pouvoir gerer le plan de version sous une gestion de configuration	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-11: Gestion des recettes		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Les recettes des modules et des outils du firmware sont enregistrées dans un outil de gestion de configuration.	
<u>Preconditions:</u>	L'utilisateur dispose d'un serveur de gestion de configuration (svn,git,mercurial). Le plan de version du firmware recense de façon exhaustive l'ensemble des recettes nécessaires à la construction du firmware.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur extrait d'un système de gestion de configuration la dernière version de la recette de construction du module busybox.</p> <pre>svn co http://svn.sagemcom.com/products/firmware A/receipes/busybox-A.txt</pre> <p>Il modifie la recette pour baser le module sur la dernière version des sources du module busybox publié sur internet puis archive cette nouvelle version de la recette dans la gestion de configuration</p> <pre>svn ci http://svn.sagemcom.com/products/firmware A/receipes/busybox-A.txt -m "reference latest busybox release"</pre>	L'utilisateur gère les différentes versions de la recette d'un module du firmware dans un système de gestion de configuration.
<u>Dernier résultat:</u>	Reussi	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	<p>L'utilisateur édite la recette du package preempt_rt pour positionner la valeur PREEMPT_RT_SITE</p> <pre>PREEMPT_RT_SITE:=http://www.kernel.org/pub/linux/kernel/projects/rt/olde r</pre> <p>dans le cas d'une version noyau 2.6.25.</p> <p>L'utilisateur enregistre cette nouvelle version de la recette dans la gestion de configuration git.</p>	
<u>Exigences fonctionnelles</u>	EX_29: Pouvoir gerer les recettes dans la gestion de configuration	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-12: Gestion des sources		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Les sources des modules sont enregistrées dans un outil de gestion de configuration.		
<u>Preconditions:</u> L'utilisateur dispose d'un serveur de gestion de configuration (svn,git,mercurial).		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur extrait d'un système de gestion de configuration la dernière version archivée du module busybox. svn co http://svn.sagemcom.com/modules/busybox Il modifie le contenu du fichier networking/dnsd.c du module et archive la nouvelle version svn ci http://svn.sagemcom.com/modules/busybox -m "add aligned attribute"	L'utilisateur gère les différentes versions des codes sources du module dans un système de gestion de configuration. Il peut consulter l'historique des versions, visualiser les différences, taguer des versions, fusionner des modifications, créer des branches.
<u>Dernier résultat:</u> Reussi		
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	L'utilisateur modifie le fichier lib/stats/Makefile du package albatros pour ajouter \$(CFLAGS) dans la règle de construction de la librairie libstats.so. Il enregistre cette nouvelle version du fichier lib/stats/Makefile dans la gestion de configuration du module albatros.	
<u>Exigences fonctionnelles</u>	EX_31: Pouvoir gerer les sources des differents composants logiciel sous gestion de configuration	
<u>mots clés:</u>	Aucun	

1.5. Test Suite : Génération

Cette suite de tests traite des outils de construction et du générateur de Firmware

Cas de Tests RTEL4I_EO-3: Plan de version d'un firmware		
<u>Auteur :</u>		Olivier GALLOT
<u>Résumé:</u> Le générateur enchaîne l'exécution des recettes du plan de version du firmware.		
<u>Preconditions:</u> Chaque firmware dispose d'un répertoire contenant son plan de version et le Makefile pour lancer sa construction. Le plan de version du firmware est la liste exhaustive des recettes des modules et des outils nécessaires à la construction du firmware. Les recettes décrivent les dépendances inter modules.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Depuis un shell unix il se place dans le repertoire firmwareA et lance la commande "make firmwareA" .	L'utilisateur dispose du fichier binaire de référence firmwareA.bin construit suivant les outils et modules décrits dans le fichier plan de version. Un fichier de trace build.log enregistre l'ensemble des étapes de compilation et d'édition de lien des binaires.
2	Dans sa copie locale du firmware l'utilisateur met à jour les sources de busybox 'svn up busybox' puis reconstruit le firmware 'make firmwareA'. La mise à jour busybox comporte des erreurs et la construction du firmware échoue.	Le fichier de trace build.log enregistre l'ensemble des étapes de compilation et d'édition de lien des binaires. L'erreur de construction du module busybox est tracée et permet de facilement situer le fichier source fautif.

<u>Dernier résultat:</u>	Echec
Session	Essais de Certification Environnement et Outils de Développement Version 1
Tester	Olivier GALLOT
Notes d'exécution	<p>Le fichier .config de buildroot n'est pas suffisant pour décrire les dépendances inter modules (ou inter packages selon la terminologie buildroot). Cf tests RTEL4I-8.</p> <p>Si le fichier .config doit être complété par un autre fichier produit par Eclipse comme évoqué alors l'implémentation du plan de version ne satisfait les exigences EX_33.2.</p> <p>Aucun fichier de trace de la construction du firmware n'est produit. Les exigences EX_36 et EX37 ne sont pas couvertes.</p>
<u>Exigences fonctionnelles</u>	<p>EX_ 1: Pouvoir construire facilement un firmware</p> <p>EX_ 18: Permettre la description exhaustive des outils et composants logiciel par le plan de version</p> <p>EX_ 32: Les sources du projet ne constituent pas une arborescence unique au sein de la gestion de configura</p> <p>EX_ 33: L'ensemble du firmware doit pouvoir être généré</p> <p>EX_ 33.1: à partir de l'IDE</p> <p>EX_ 33.2: à partir d'une commande en ligne</p> <p>EX_ 35: La génération doit s'arrêter sur une erreur bloquante</p> <p>EX_ 36: l'outil doit aider l'utilisateur à exploiter les erreurs de générations</p> <p>EX_ 37: l'outil doit pouvoir fabriquer un fichier de log des erreurs</p> <p>EX_ 4: Avoir un outil insensible à l'environnement (l'usage de technique comme chroot est encouragée).</p> <p>EX_ 6: Pouvoir supporter les ARM11</p> <p>EX_ 9: Pouvoir supporter les SH4</p>
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-5: Profils du firmware		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Le générateur construit à la demande de l'utilisateur l'ensemble des profils ou un profil désigné du firmware.		
<u>Preconditions:</u> Le firmware dispose de plusieurs profils. Par défaut le générateur construit tous les profils déclarés dans le plan de version du firmware.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Depuis un shell unix il se place dans le repertoire firmwareA et lance la construction de tous les profils avec la seule commande "make firmwareA" .	Pour chaque profil défini dans le plan de version du firmware, un espace de génération qui lui est propre recueille le binaire du firmware et le jeu complet des binaires qui le composent. Les espaces et les binaires des différents profils coexistent de façon indépendante et peuvent être générés simultanément (make -j 16).
2	L'utilisateur instrumente le code d'un module du firmware puis génère le profil debug du firmware 'make debug-firmwareA'	Seuls les binaires dans l'espace du profil debug sont reconstruits.
<u>Dernier résultat:</u>	Echec	
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	Le générateur buildroot ne permet pas de spécifier différents profils pour un firmware donné.	
<u>Exigences fonctionnelles</u>	EX_ 24: Pouvoir decrire plusieurs profils d'un soft EX_ 24.5: Pouvoir exploiter simultanement sur une meme machine plusieurs profils sans avoir de conflit. EX_ 3: Pouvoir facilement mettre en Suvre de facon concurrente plusieurs profils de la meme version.	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-7: Génération unitaire d'un outil		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Générer de façon unitaire une version spécifique d'un outil pour le firmware désigné.	
<u>Preconditions:</u>	L'outil et sa version spécifique sont référencés dans le plan de version du firmware.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Il édite le plan de version et enregistre l'utilisation de gcc version 4.6 comme compilateur. Depuis un shell unix il se place dans le repertoire firmwareA et lance la construction du cross compilateur gcc version 4.6	Le binaire gcc version 4.6 cross compilé pour la plateforme cible est disponible sous l'arborescence firmwareA
2	L'utilisateur lance la génération complète du firmware avec cette version 4.6 de gcc.	Le firmware cible est construit avec la dernière version de gcc.
<u>Dernier résultat:</u>	Non Disponible	
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	Le firmware est construit avec une toolchain externe	
<u>Exigences fonctionnelles</u>	EX_ 19: Permettre de decire strictement dans le plan de version les noms et les versions logicielles de ch EX_ 21: Pouvoir modifier de facon dynamique les divers elements du plan de version EX_ 21.1: Pouvoir ajouter, retirer, modifier un outil EX_ 30: Pouvoir gerer les outils sous gestion de configuration	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-8: Re-génération suite à modification		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Toute modification d'un élément constitutif du firmware induit la reconstruction du binaire du firmware.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur applique un patch spécifique pour sa plateforme au code source de la version 4.6 de gcc puis relance la génération du firmware.	L'ensemble des binaires constituant le firmware sont recompilés.
2	L'utilisateur modifie la recette de gcc pour qu'il soit généré avec la librairie uclibc plutôt que la glibc puis relance la génération du firmware.	L'ensemble des binaires constituant le firmware sont recompilés.
3	L'utilisateur applique un patch au code source du module busybox embarqué puis régénère le firmware.	Seuls les binaires du module busybox et celui du firmware sont régénérés.
4	L'utilisateur modifie la recette du module busybox pour référencer le code source de la version antérieur à la dernière puis relance la construction du firmware.	Seuls les binaires du module busybox et celui du firmware sont régénérés.
5	L'utilisateur édite le plan de version et change le nom du firmware puis relance sa génération.	Seuls le binaire embarquant le nom du firmware et le firmware lui même sont reconstruits.
6	L'utilisateur édite le plan de version et ajoute une option de compilation pour optimiser la taille des binaires produits: -Os.	L'ensemble des binaires constituant le firmware sont recompilés.
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	L'utilisateur construit une première fois le firmware désigné par le pla de version sodimm2410_glibc_2625_xeno_sourcery_defconfig Ensuite il ajoute les paramètres suivants au plan de version sodimm2410_glibc_2625_xeno_sourcery_defconfig: BR2_ENABLE_DEBUG=y BR2_DEBUG_3=y BR2_PACKAGE_GDB_SERVER=y et relance la génération du firmware.	

	<p>Seul le binaire gdbserver est construit puis embarqué (output/target/usr/bin/gdbserver). Le générateur impose la version gdbserver présente dans le générateur buildroot/toolchain. Malgré l'utilisation d'une toolchain externe pour la carte sodimm2410, il n'est pas possible de configurer buildroot pour qu'il embarque la version gdbserver de cette toolchain externe.</p> <p>L'erreur la plus gênante est que le générateur n'embarque pas du tout la librairie libthread_db.so et ne recompile pas l'ensemble des binaires en -g3 comme demandé (BR2_DEBUG_3).</p> <p>Les exigences EX_21 et EX_21.3 ne sont pas couvertes.</p>
<p><u>Exigences fonctionnelles</u></p>	<p>EX_19: Permettre de decrire strictement dans le plan de version les noms et les versions logicielles de ch</p> <p>EX_21: Pouvoir modifier de facon dynamique les divers elements du plan de version</p> <p>EX_21.2: Pouvoir ajouter, retirer, modifier un des composants logiciel du firmware</p> <p>EX_21.3: Ne pas avoir a tout systematiquement re-generer lorsque le plan de version est modifie</p>
<p><u>mots clés:</u></p>	<p>Aucun</p>

Cas de Tests RTEL4I_EO-9: Génération unitaire d'un module		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Générer de façon unitaire un module du firmware désigné.	
<u>Preconditions:</u>	Le module est référencé dans le plan de version du firmware.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Depuis un shell unix il se place dans le repertoire firmwareA et lance la génération du module busybox avec la commande "make busybox" .	Seuls les binaires des outils de cross compilation et du module busybox sont générés. Un fichier de traces enregistre les commandes de compilation et d'édition de lien de tous les binaires construits.
2	L'utilisateur exécute les actions du point précédent dans l'IDE.	Les résultats sont identiques au point précédent.
<u>Dernier résultat:</u>	Reussi	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Les exigences EX_34 et EX_34.2 sont couvertes mais la construction du module est enregistrée dans aucun fichier de trace.	
<u>Exigences fonctionnelles</u>	EX_ 21: Pouvoir modifier de façon dynamique les divers éléments du plan de version EX_ 21.2: Pouvoir ajouter, retirer, modifier un des composants logiciel du firmware EX_ 21.3: Ne pas avoir à tout systématiquement re-générer lorsque le plan de version est modifié EX_ 34: Un outil ou un composant logiciel doit pouvoir être compilé de façon unitaire EX_ 34.1: à partir de l'IDE. EX_ 34.2: à partir d'une commande en ligne.	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-13: Composition variable du firmware		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Les profils d'un firmware référencent des compositions distinctes.		
<u>Preconditions:</u> Les profils 'release', 'debug' et 'profiling' du firmware sont définis.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	Le profil debug embarque gdbserver en sus des binaires embarqués par le profil release.	Le firmware de profil debug permet de réaliser un session de debug.
2	Le profil profiling configure le noyau avec oprofile, embarque les binaires opcontrol et oreport.	Le firmware permet de tracer les codes qui occupent le plus de temps processeur
<u>Dernier résultat:</u> Echec		
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le générateur buildroot ne permet pas de spécifier différents profils pour un firmware donné.	
<u>Exigences fonctionnelles</u>	EX_ 24.1: pouvoir invalider certain soft dans un profil	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-14: Option spécifique à un profil		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Une option attribuée à un profil spécifique ne perturbe pas la génération des autres profils.		
<u>Preconditions:</u> Le firmware dispose de plusieurs profils		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur édite le plan de version du firmware et affecte au profil debug les options de compilation -g3 -O2 alors qu'elles sont positionnées à -Os pour le profil release.	Les binaires produits dans l'espace de génération du profil debug embarquent les informations de debug. Les binaires produits dans l'espace de génération du profil release n'embarquent pas d'information de debug et sont optimisés en taille.
<u>Dernier résultat:</u> Echec		
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	Le générateur buildroot ne permet pas de spécifier différents profils pour un firmware donné.	
<u>Exigences fonctionnelles</u>	EX_24.3: Pouvoir passer des options a tout les modules et outils d'un profil	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-15: Re-génération unitaire d'un module		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Le générateur reconstruit uniquement le module désigné.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur modifie la recette du module busybox pour référencer le code source de la version antérieure à la dernière puis relance la construction du module.	Seuls les binaires du module busybox sont reconstruits. Le binaire du firmware cible n'est pas reconstruit.
2	L'utilisateur applique un patch au code source du module busybox embarqué puis régénère le module.	Seuls les binaires du module busybox sont reconstruits. Le binaire du firmware cible n'est pas reconstruit.
<u>Dernier résultat:</u> Echec		
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	1. Pas de re-génération suite à l'application d'un patch du code source de output/build/busybox/shell/ash.c 2. Pas de re-génération suite à l'application d'un patch du makefile package/busybox/busybox.mk	
<u>Exigences fonctionnelles</u>	EX_39: Pouvoir relancer une generation de module en ne compilant que ce qui est strictement necessaire	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-16: Re-génération du firmware		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Le firmware est re-généré suite à une modification des sources d'un module.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur extrait les sources du module librairie polarssl, applique un patch puis relance la construction du firmware.	La librairie polarssl est reconstruite ainsi que tous les binaires qui la lie statiquement. L'ensemble des binaires et des actions qui dépendent de ces binaires nouvellement reconstruits sont réexaminés et reconstruits à leur tour jusqu'à produire le nouveau firmware. Les binaires qui n'ont pas de dépendance avec ceux nouvellement reconstruits ne sont pas recompilés. Le binaire obtenu est équivalent à celui qui aurait été produit à partir de l'arborescence du firmware vierge de toute compilation.
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le firmware n'est pas re-généré suite à l'application d'un patch du code source de output/build/busybox/shell/ash.c	
<u>Exigences fonctionnelles</u>	EX_38: Pouvoir relancer une generation de firmware en ne compilant que ce qui est strictement necessaire	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-17: Génération parallélisée		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	La génération du firmware est distribuée sur plusieurs jobs.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Depuis un shell unix il se place dans le repertoire firmwareA et lance la commande "make firmwareA -j 16" pour exploiter au mieux la puissance de sa machine de travail.</p>	<p>Le générateur derive plusieurs "job unix" pour construire les modules du firmware "en parallèle" et accélérer significativement le temps de production du binaire du firmware. Les "jobs" sont distribués sur le système Linux en général mais pas nécessairement sur des processeurs distincts. Le générateur respecte les dépendances explicites et implicites des binaires avec les codes sources ainsi que les dépendances inter modules renseignées dans les recettes afin de garantir l'intégrité et la reproductibilité de la construction du firmware.</p>
<u>Dernier résultat:</u>	Reussi	
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	L'utilisateur édite le plan de version et renseigne le champs BR2_JLEVEL pour spécifier le nombre de jobs unix que le générateur pourra lancer afin de construire plusieurs cibles en parallèle et accélérer la construction du firmware.	
<u>Exigences fonctionnelles</u>	EX_40: rapidite : il doit etre possible d'exploiter les processeurs multi-cores pour acclerer la generati	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-19: Recette		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Toutes les techniques de génération des modules ou outils logiciels employés par la communauté Open Sources peuvent être mises en oeuvre dans les recettes du générateur.		
<u>Preconditions:</u> Le générateur construit les binaires d'un outil ou d'un module du firmware selon les méthodes décrite dans la recette.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur intègre au plan de version du firmware un module construit avec CMake.	Les binaires du module sont produits par une commande du générateur de firmware par le moteur CMake.
2	L'utilisateur intègre au plan de version du firmware un module construit avec sCons.	Les binaires du module sont produits avec une commande du générateur de firmware par le moteur sCons.
3	L'utilisateur intègre au plan de version du firmware un module décrit suivant Kbuild	Les binaires du module sont produits avec une commande du générateur de firmware.
4	L'utilisateur intègre au plan de version du firmware un module configuré avec le système Autotools.	Les binaires du module sont produits avec une commande du générateur de firmware qui commandera à son tour la configuration des sources avec les Autotools puis leur compilation.
5	L'utilisateur intègre au plan de version du firmware un module construit avec Make.	Les binaires du module sont produits avec une commande du générateur de firmware par le moteur Make.
6	L'utilisateur intègre au plan de version du firmware un module construit avec gmake.	Les binaires du module sont produits avec une commande du générateur de firmware par le moteur gmake.
<u>Dernier résultat:</u> Reussi		
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	L'utilisateur embarque les utilitaires du package cdrkit dont la recette est écrite suivant CMake: BR2_PACKAGE_CDRKIT=y	

	Les binaires sont construits et intégrés au firmware.
<u>Exigences fonctionnelles</u>	EX_ 25: Pouvoir ajouter ses propres recettes EX_ 26: Pouvoir construire les recettes de façon à pouvoir aborder les techniques de construction les plus
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-35: module avec option spécifique à un profil		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Générer le module d'un profil firmware avec une option spécifique au profil désigné.	
<u>Preconditions:</u>	Le firmware dispose de plusieurs profils	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur édite la recette du module busybox, ajoute l'option de compilation -O0 pour le profil debug en remplacement de la valeur par défaut -O2 et génère le module depuis la commande du générateur pour les profils debug et release.	Le binaire busybox du profil debug est généré sans optimisation alors que celui du profil release est compilé avec l'option -O2.
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le générateur buildroot ne permet pas de spécifier différents profils pour un firmware donné.	
<u>Exigences fonctionnelles</u>	EX_ 24.2: Pouvoir passer des options spécifiques à certains modules dans le cadre d'un profil	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-36: Option générique pour tous les profils		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Désigner une option prise en compte par tous les profils	
<u>Preconditions:</u>	Le firmware dispose de plusieurs profils	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur édite le plan de version du firmware, ajoute la directive de compilation <i>-msoft-float</i> destinée à tous les profils et lance la génération de tous les profils du firmware.	Tous les binaires de tous les profils du firmware sont générés sans utiliser les instructions de Floating Point Unit (FPU).
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Le générateur buildroot ne permet pas de spécifier différents profils pour un firmware donné.	
<u>Exigences fonctionnelles</u>	EX_ 24.4: Pouvoir passer des options a tout les modules et outils de tout les profils	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-39: Nettoyage complet du firmware		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Revenir facilement aux conditions initiales de la construction du firmware.	
<u>Preconditions:</u>	Le firmware dispose de plusieurs profils	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Avec une commande du générateur il lance la construction du firmwareA . A l'issue de la construction du firmware, l'utilisateur invoque la commande du générateur pour nettoyer l'ensemble des production du firmwareA.	A tout moment et simplement, le développeur peut effacer l'ensemble des fichiers générés lors de la construction du firmware: L'arborescence firmwareA est nettoyée de tous les fichiers générés.
2	L'utilisateur reprend le point précédent mais demande cette fois ci de ne nettoyer que le profil debug du firmware.	L'ensemble des fichiers produits pour le profil debug sont effacés.
<u>Dernier résultat:</u>	Reussi	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	rm -rf output	
<u>Exigences fonctionnelles</u>	EX_ 41: Pouvoir effacer simplement tous les fichiers generes lors de la generation	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-40: Nettoyage d'un module du firmware		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u> Revenir facilement aux conditions initiales de la construction d'un module du firmware.		
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Avec une commande du générateur il lance la construction du firmwareA . A l'issue de la construction du firmware, l'utilisateur invoque une commande du générateur pour nettoyer le module busybox.	L'ensemble des fichiers générés lors de la construction du module busybox sont effacés.
2	L'utilisateur reprend le point précédent mais demande cette fois ci de nettoyer le profil debug du module busybox.	Seuls les fichiers générés lors de la construction du module busybox dans le cadre du profil debug sont effacés.
<u>Dernier résultat:</u> Reussi		
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	L'utilisateur efface le module albatros rm -rf output/build/albtros-1.0	
<u>Exigences fonctionnelles</u>	EX_42: Pouvoir effacer simplement tous les fichiers generes d'un module	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-41: Nettoyage du système de fichiers cible du firmware		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Nettoyer le système de fichiers cible sans reconstruire le firmware en entier.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	<p>L'utilisateur récupère le firmware enregistré dans un système de gestion de configuration: svn co http://svn.sagemcom.com/products/firmwareA Avec une commande du générateur il lance la construction du firmwareA . A l'issue de la construction du firmware, l'utilisateur invoque la commande du générateur pour nettoyer le système de fichiers du firmwareA. Il édite le plan de version du firmware, retire un module et reconstruit le firmware.</p>	<p>Le firmware complet est généré un première fois. Le système de fichiers est effacé suite à l'action de nettoyage puis reconstruit sans embarquer les binaires du module retiré du plan de version.</p>
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	<p>L'utilisateur construit une première fois le firmware. Il efface le système de fichier cible <code>rm -rf output/target/ output/images/rootfs.*</code> modifie le plan de version afin de ne plus embarquer les binaires du module IPROUTE2: BR2_PACKAGE_IPROUTE2 is not set puis lance enfin la construction du nouveau firmware.</p> <p>La construction échoue avec l'erreur suivante:</p> <pre>echo "Welcome to Buildroot (SODIMM2410/Glibc)" > /home/user/sodimm2410/buildroot-ow/output/target/etc/issue /bin/sed -i -e '# GENERIC_SERIAL\$/s~^.*#~ttySAC0::respawn:/sbin/getty -L ttySAC0 115200 vt100 #~' \ /home/user/sodimm2410/buildroot-ow/output/target/etc/inittab /bin/sed: can't read /home/user/sodimm2410/buildroot- ow/output/target/etc/inittab: No such file or directory make: *** [target-generic-getty] Error 2</pre>	
<u>Exigences fonctionnelles</u>	EX_ 43: Pouvoir effacer simplement tous les fichiers installés dans le pseudo filesystem de sortie	
<u>mots clés:</u>	Aucun	

1.6. Test Suite : mesure

Mesurer les performances du système

Cas de Tests RTEL4I_EO-30: mesure de la mémoire		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Présentation de l'utilisation mémoire de chaque processus en exécution dans le système.	
<u>Preconditions:</u>	La mesure peut être réalisée lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur ajoute le module htop au plan de version du firmware, génère le binaire du firmware et exécute htop sur la cible.	L'utilisateur visualise dans htop l'occupation mémoire de chaque processus en exécution dans le système.
2	L'utilisateur construit le firmware et exécute l'outil système vmstat	L'utilisateur visualise dans vmstat l'occupation mémoire de chaque processus en exécution dans le système.
3	L'utilisateur embarque le programme pmap dans le firmware et l'exécute sur cible.	le programme pmap présente l'occupation mémoire de chaque processus en exécution dans le système.
4	L'utilisateur embarque le programme smaps dans le firmware et l'exécute sur cible.	Le programme smaps présente l'occupation mémoire de chaque processus en exécution dans le système.
<u>Dernier résultat:</u>	Reussi	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	<p>L'utilisateur embarque les utilitaires du module procps: vmstat, top, pmap BR2_PACKAGE_PROCP=y dans le firmware et trace l'occupation mémoire de l'application signal_stream d'albtros.</p> <p>L'utilitaire top présente l'activité de l'application signal_stream mais pas le détail pour chacun des threads de l'application.</p> <p>Les modules htop et atop permettraient de donner plus de détails.</p>	

<u>Exigences fonctionnelles</u>	EX_ 54: Pouvoir mesurer l'utilisation de la memoire
<u>mots clés:</u>	Aucun

Cas de Tests RTEL4I_EO-31: mesure de latence et gigue		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Présenter les latences et gigue pour l'exécution d'un ensemble de codes désignés.	
<u>Preconditions:</u>	Les mesures de latences et gigue peuvent être réalisées lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur embarque l'outil latencyTop dans le firmware et l'exécute sur la cible.	L'outil présente les points de latence pour le système et par processus en exécution dans le système.
2	L'utilisateur embarque le test https://rt.wiki.kernel.org/index.php/Cyclictest dans le firmware et l'exécute sur la cible.	Le test cyclictest est utilisable sans restriction
<u>Dernier résultat:</u>	Echec	
<u>Session</u>	Essais de Certification Environnement et Outils de Développement Version 1	
<u>Tester</u>	Olivier GALLOT	
<u>Notes d'exécution</u>	Le module latencytop n'est pas intégré au générateur buildroot.	
<u>Exigences fonctionnelles</u>	EX_ 55: Pouvoir faire des mesures de gigue et latence	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-32: mesure de la charge du système		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Présenter l'utilisation processeur du système.	
<u>Preconditions:</u>	Les mesures d'utilisation processeur peuvent être réalisées lors d'une exécution de l'application de traitement de flux de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur embarque l'outil http://freshmeat.net/projects/atop dans le firmware et l'exécute sur la cible.	L'outil atop présente l'utilisation processeur de tous les processus en exécution dans le système.
2	L'utilisateur embarque l'outil http://htop.sourceforge.net dans le firmware et l'exécute sur la cible.	L'outil htop présente l'utilisation processeur du système pour tous les threads des processus en exécution dans le système.
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	Aucun des modules atop et htop est embarqué dans le générateur de firmware buildroot.	
<u>Exigences fonctionnelles</u>	EX_ 56: Pouvoir mesurer la charge du CPU	
<u>mots clés:</u>	Aucun	

Cas de Tests RTEL4I_EO-33: mesure de débit		
<u>Auteur :</u>	Olivier GALLOT	
<u>Résumé:</u>	Mesurer un flux au sein du système.	
<u>Preconditions:</u>	Les mesures de débit peuvent être réalisées lors d'une exécution de l'application type VOIP de la suite de tests temps réel albatros.	
<u>#:</u>	<u>Step actions:</u>	<u>Résultats attendus:</u>
1	L'utilisateur embarque les outils iperf, ifstat et vnStat dans le firmware et exécute tour à tour chacun d'eux.	Chacun des outils mesure et présente les débits des flux qui traversent le système.
<u>Dernier résultat:</u>	Echec	
Session	Essais de Certification Environnement et Outils de Développement Version 1	
Tester	Olivier GALLOT	
Notes d'exécution	<p>L'utilisateur embarque les outils des module iproute2 et iperf dans le firmware.</p> <p>Il ajoute BR2_PACKAGE_IPERF=y et BR2_PACKAGE_IPROUTE2=y au plan de version.</p> <p>La compilation de iperf échoue avec le message d'erreur suivant: iperf did not compiled because of configure: error: C++ compiler cannot create executables</p>	
<u>Exigences fonctionnelles</u>	EX_ 57: Pouvoir mesurer le debit au niveau des interfaces	
<u>mots clés:</u>	Aucun	

Synthèse des résultats

Résultats par suites de tests de haut niveau

Suite	Total	Non exécuté	Reussi	Echec	Bloqué	Non Disponible	Inconnu	Achévé [%]
Analyse	5	0	0	4	0	1	0	100.0
Debug	6	0	1	2	0	3	0	100.0
Edition	4	0	1	1	0	2	0	100.0
Gestion de Configuration	3	0	3	0	0	0	0	100.0
Génération	16	0	5	10	0	1	0	100.0
mesure	4	0	1	3	0	0	0	100.0

Résultats par Plate-forme

Plate-forme	Total	Non exécuté	[%]	Reussi	[%]	Echec	[%]	Bloqué	[%]	Non Disponible	[%]	Inconnu	[%]	Achévé [%]
SODIMM2410	38	0	0.00	11	28.95	20	52.63	0	0.00	7	18.42	0	0.00	100.00