CSE - Conception des systèmes embarqués

Aspects liés au logiciel dans les systèmes embarqués

Prof. Daniel Rossier

Aspects logiciels dans l'embarqué

- Le *software* dans les systèmes embarqués
- Systèmes d'exploitation pour l'embarqué
- Aspects liés au design de logiciel embarqué

Informatique embarquée

- **Les contraintes liées à l'informatique embarquée:**
  - contrainte de taille mémoire
  - contrainte de temps de réponse
  - contrainte de fiabilité
  - contrainte de sécurité
  - contrainte de ressource d'énergie/autonomie
  - contrainte d'architecture matérielle
  - contrainte de prix de développement (pur et licences)
  - contrainte de prix de vente (amortissement et royalties)
  - contraintes... juridiques !

- **Le logiciel libre**
  - Disponibilité du code source
  - Possibilité de réaliser des travaux dérivés
  - Redistribution sans royalties

Langages (1/3)

- **Assembleur**
  - **Bas niveau**, structures de programme élémentaires
  - **Forte dépendance au matériel** (microcontrôleur/DSP)
    - Peu d'abstraction au niveau du codage
    - Difficile à coder
  - **Performance élevée**
    - Code *profilé* au mieux pour un type de matériel

- **Exemples**
  - ARM, MIPS, PowerPC (IA32/64), SPARC, etc.

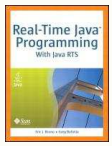
- **Utilisation**
  - ***Debug***
  - Amorçage (*Bootstrap*), *Runtime, accès I/O*, etc.
  - Instructions spécifiques (chiffrement, mathématiques, etc.) / Optimisation

Langages (2/3)

- *C*
  - **Haut niveau**
  - Structures de programmation de haut niveau (boucles *for*<>, *while*<>, tests conditionnels, types et structures, etc.)
  - Largement **répandu** dans l'industrie
  - Très proche du **bas niveau** (assembleur)
    - Accès aux adresses mémoire (pointeurs)
  - **Risques élevés de *bugs*** et absence de portabilité
- **Standards C**
  - **C89 / C90 (ANSI)**
  - C99 (ANSI/ISO/IEC 9899:1999)
  - C11 (2011)

Langages (3/3)

- **Langages OO** (Orienté-objet)
  - Le paradigme de programmation OO permet de modéliser une application à un haut niveau d'abstraction.
  - Notion de réutilisabilité et de portage
- *C++*
  - Extension du C pour le support de l'OO
  - Très utilisé dans l'industrie, surtout avec les interfaces graphiques (GUI)
- *Java*
  - Très utilisé dans l'embarqué
    - Enormément de composants réutilisables dans le domaine des protocoles (telco, multimedia, …)
  - A beaucoup évolué grâce à Internet et aux télécommunications
  - Java RTS (*Realtime System*)

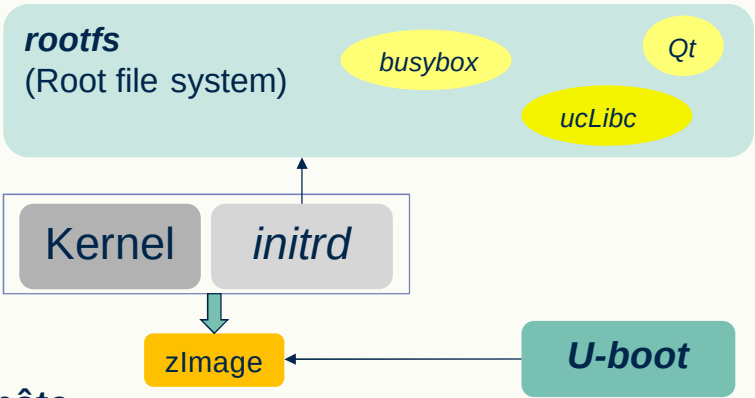


Environnement embarqué (1/2)

- ***IDE (Integrated Development Environment)***
- ***Open Source***
  - IDE **Eclipse**, *Topcased*
  - *Arctic Core, Arctic Studio*
  - *Toolchains* **GNU** (*gcc, gas, ld, etc.*)
- ***Commercial***
  - ***Visual Studio***
  - *HP LabView,* ***Mathworks*** *(Matlab, ...)*
  - *Luminosity IDE*
  - *Integrity (Green Hills)*
  - *etc.*

Environnement embarqué (2/2)

- **Environnement cible**



- **Environnement hôte**

IDE Eclipse

git, svn, …

(cross-)Toolchain
- compiler - readelf
- linker - objdump
- gdb - nm

picocom
minicom
Hyperterminal

Utilities
- grep, find, awk, phyton ..

Framework pour l'embarqué (1/6)

- **Une histoire de *toolchain…***
  - Règle d'or: d'une manière générale, utiliser la *toolchain* fourni avec le BSP !

- ***Crosstool***
  - ***http://kegel.com/crosstool/***
- **Construction de *cross-toolchains***
  - **Permet un profilage très spécifique par rapport à une plate-forme**
- ***Toolchains* pré-compilés**
  - ***CodeSourcery***
  - ***http://elinux.org/ARMCompilers***

Framework pour l'embarqué (2/6)

- ***OpenEmbedded***
  - *http://www.openembedded.org*

- Beaucoup de **plates-formes** et de ***releases*** supportées
- Plusieurs milliers de *packages* pouvant être construits (GTK+, Qt, X Windows, Mono, Jave, etc.)
- Outils de génération de *toolchains*, d'image noyau, de systèmes de fichiers, d'applications, etc.
- Plusieurs ***Mailing-lists*** très actives, très bon support
- Fondé sur ***BitBake***
  - Ensemble d'outils (la plupart écrits en ***Python***)
  - Dérivé de *Portage* (gestion de paquet sur *Gentoo*)
  - Définition de *méta-données* pour la configuration d'un système

Framework pour l'embarqué (3/6)

- **Poky**
  - *http://www.pokylinux.org*

- **Fondé sur *OpenEmbedded***
  - **Produit un BSP ciblé sur certaines technologies**
    - **Linux, X11, Matchbox, GTK+, Pimlico, Clutter, utilitaires GNOME Mobile**
    - **Orienté x86 et ARM principalement**
  - **Bon support avec QEMU**

- **Environnement graphique SATO**
  - **Taille d'écran réduite**

Framework pour l'embarqué (4/6)

- **Yocto**
  - *http://www.yoctoproject.org*

- **"The Yocto Project™ is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture."**

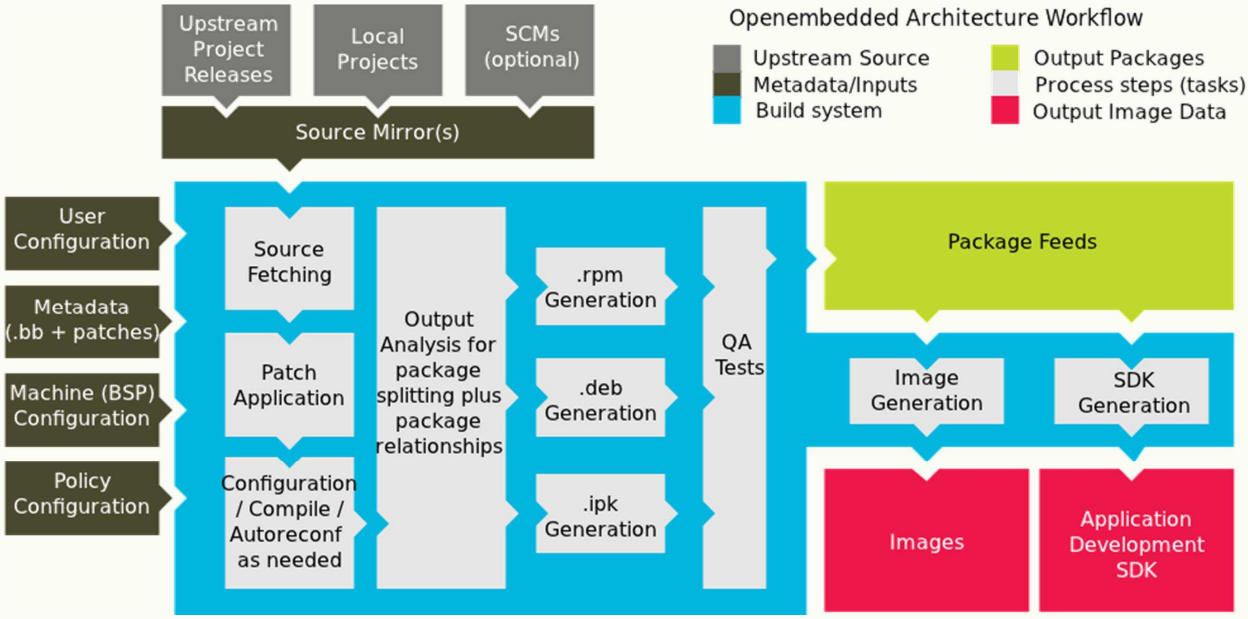
- **Fusion de plusieurs approches**
  - ***OpenEmbedded, Poky, Bitbake, etc.***

- **Très orienté *Python***

- **Construction d'un BSP complet**
  - ***Version minimale ou plus complète***

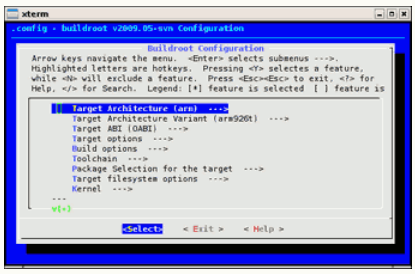
Framework pour l'embarqué (5/6)

- **Environnement Yocto**



Framework pour l'embarqué (6/6)

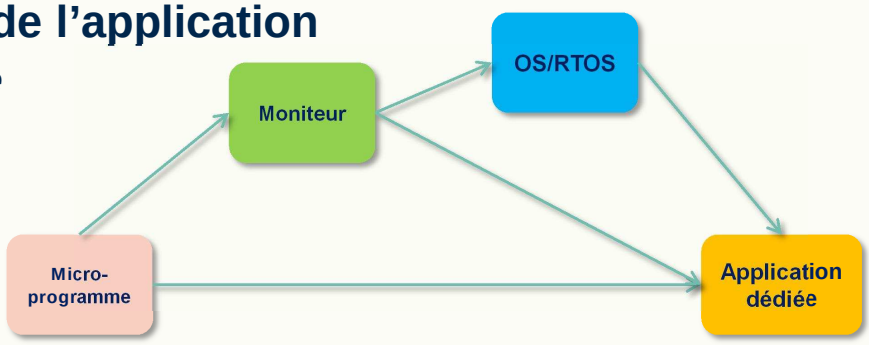
- **Buildroot**
  - **Ensemble de scripts pour la génération d'une *toolchain,* noyau, *rootfs*, et *bootloader***



- **OpenWrt**
  - **Génère un *firmware* pour des composants réseau (routeurs, gateways, etc.)**
  - **Orienté réseau avec beaucoup de fichiers de configuration pré-définis (routage, NAT, etc.)**
- **Autres: *LTIB, T2 SDE, PTXdist, uClinux-dist, Denx ELDK, etc.***

Environnements d'exécution (1/2)

- **Microprogramme de démarrage**
  - **Séquence de *bootstrap* (code d'amorçage)**
- **Moniteur, BIOS**
  - **Configurations initiales**
  - **Auto-détection du matériel (recensement)**
- **Démarrage de l'application**
  - ***Standalone***
  - **OS**



Environnements d'exécution (2/2)

- **Moniteurs ou *bootloaders* ou chargeurs d'amorçage ou moniteur de *boot*…pour l'embarqué**
  - **U-boot**
  - Micromonitor
  - **RedBoot**
  - YAMON
  - LinuxBIOS
  - Lilo
  - Grub
  - **…**

U-Boot

- **Bootstrap (ARM926EJS)**

```
.globl _start
_start:
                b               reset
                ldr             pc, _undefined_instruction
                ldr             pc, _software_interrupt
                ldr             pc, _prefetch_abort
                ldr             pc, _data_abort
                ldr             pc, _not_used
                ldr             pc, _irq
                ldr             pc, _fiq
...

reset:
                /*
                 * set the cpu to SVC32 mode
                 */
                mrs             r0,cpsr
                bic             r0,r0,#0x1f
                orr             r0,r0,#0xd3
                msr             cpsr,r0

                bl              cpu_init_crit

relocate:                                               /* relocate U-Boot to RAM         */
                adr             r0, _start              /* r0 <- current position of code   */
                ldr             r1, _TEXT_BASE          /* test if we run from flash or RAM */
                cmp              r0, r1                 /* don't reloc during debug         */
                beq              stack_setup

                ldr             r2, _armboot_start
                ldr             r3, _bss_start
                sub             r2, r3, r2              /* r2 <- size of armboot            */
                add             r2, r0, r2              /* r2 <- source end address         */

copy_loop:
                ldmia           r0!, {r3-r10}           /* copy from source address [r0]    */
                stmia           r1!, {r3-r10}           /* copy to   target address [r1]    */
                cmp             r0, r2                                  /* until source end addreee [r2]    */
                ble             copy_loop
```

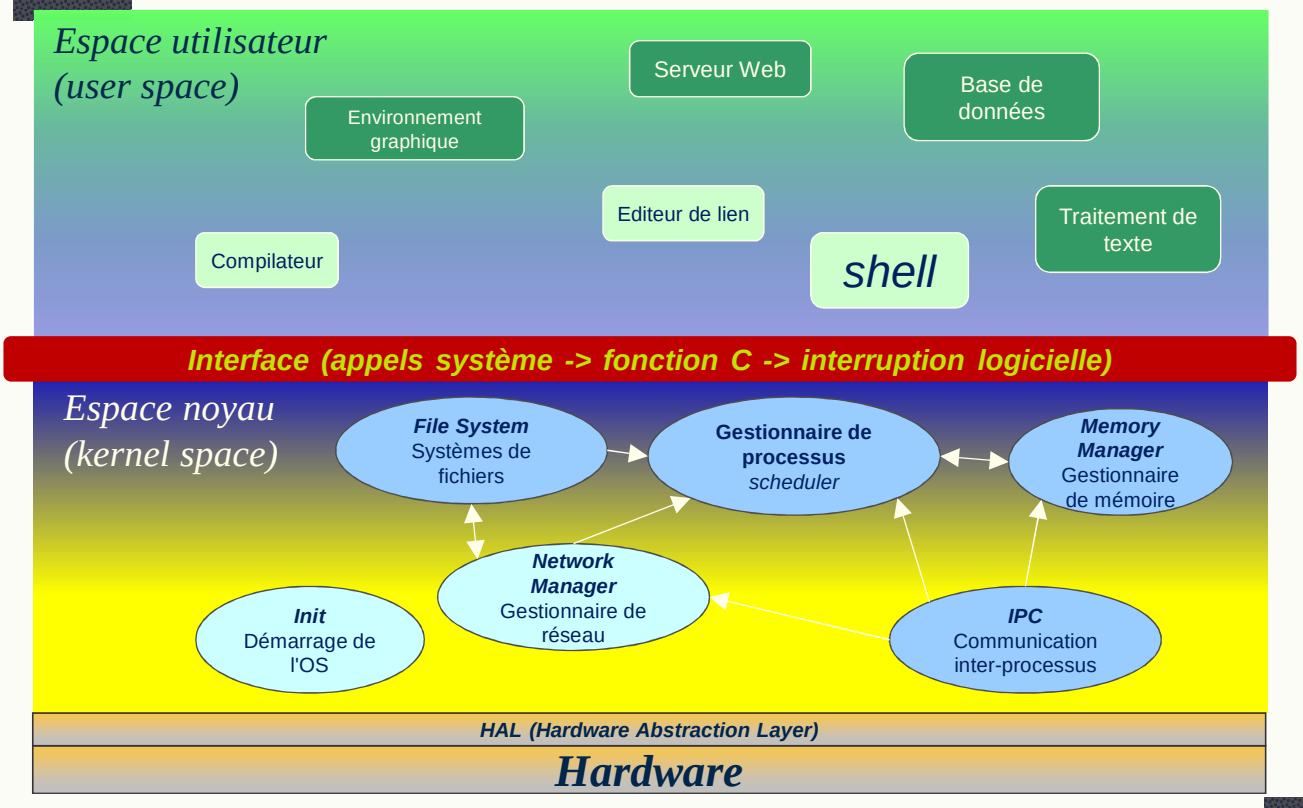

Aspects logiciels dans l'embarqué

- Le *software* dans les systèmes embarqués
- **Systèmes d'exploitation pour l'embarqué**
- Aspects liés au design de logiciel embarqué

Systèmes d'exploitation (1/3)

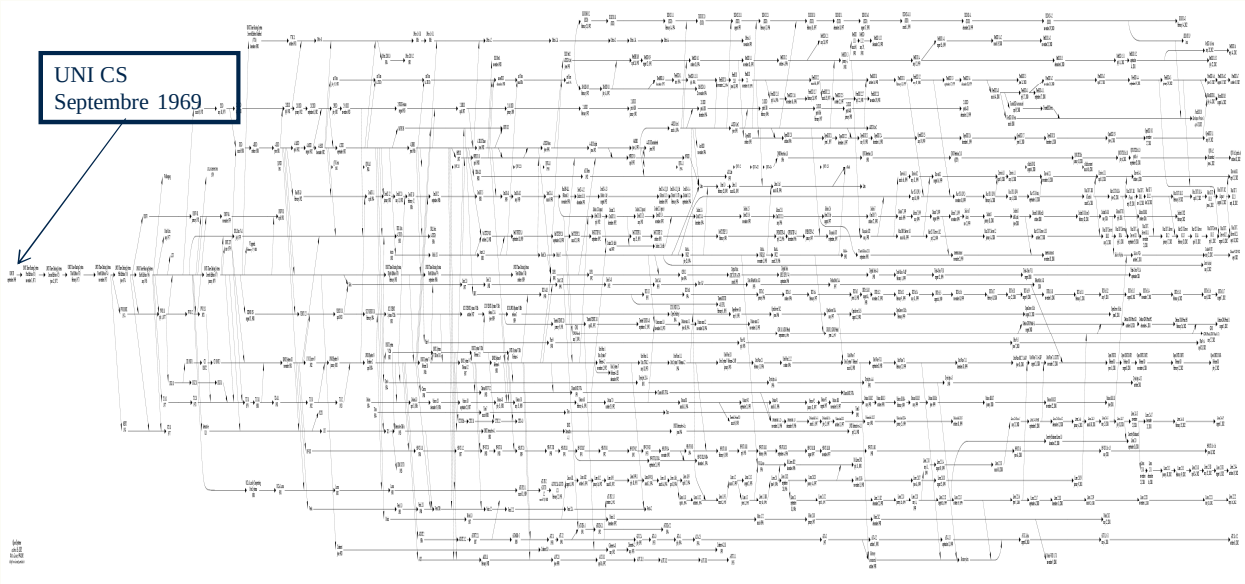
- **Pourquoi un OS dans l'embarqué?**
  - Gestion de la complexité d'une application au travers de *facilités (*ou *services*)
    - Abstraction de la complexité logicielle
  - Accès optimisé aux ressources externes via des I/Os
    - Abstraction de la complexité matérielle

Systèmes d'exploitation (2/3)



Systèmes d'exploitation (3/3)

- http://www.levenez.com/windows Généalogie de Windows
  http://www.levenez.com/unix Généalogie d'Unix

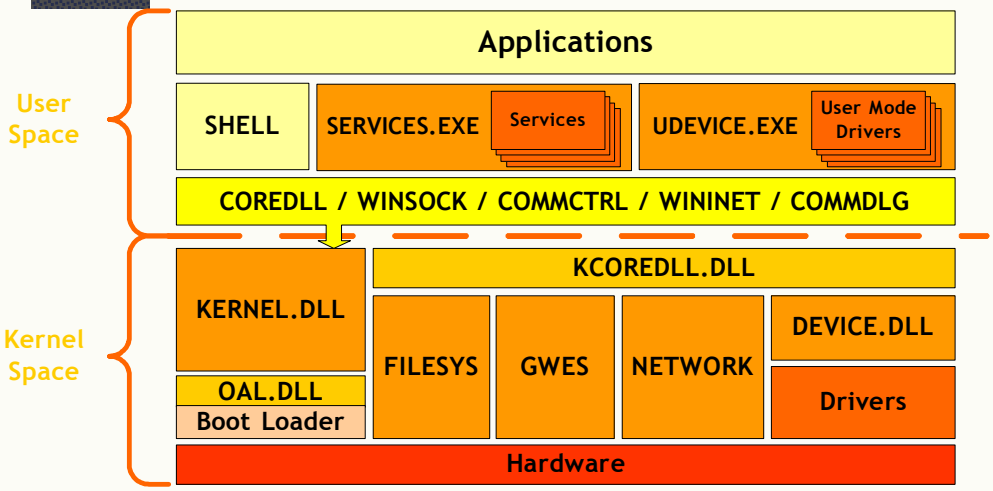


OS embarqués

- Microsoft
  - Windows Mobile (Windows CE)
    - Noyau et composants légers (taille image: ~300ko !)
    - Peut supporter du temps-réel strict
    - Seules les fonctionnalités nécessaires sont présentes
    - => Windows Mobile
      - Fonctionnalités pour téléphones mobiles
    - Windows Embedded Compact 7

  - Windows Embedded 7
    - Anciennement Windows XP Embedded
    - Beaucoup de *drivers*
    - Développement facile
    - Nécessite plus de place

Windows Embedded (CE)

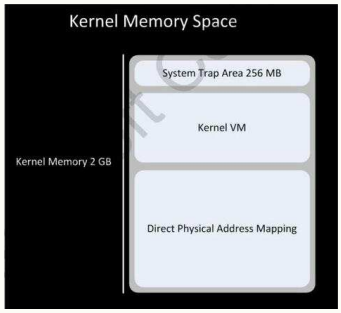


- WinCE 5.0:
  - 32 Mo d'adresses (virtuelles) par processus
- WinCE 6.0:
  - 4 Go d'adresses (virtuelles) par processus (1 Go par processus utilisateur)

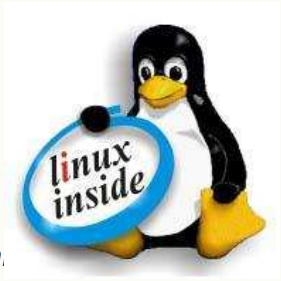
Windows Phone 7, 8

- Rupture de compatibilité avec Windows Mobile 6.5
- Conçu pour supporter les écrans tactiles capacitifs
- Séparation 2G:2G

- Difficile d'avoir des infos sur l'architecture interne, mais apparemment très proche des versions précédentes.
  - Les APIs utilisateurs sont différentes.



Linux (1/4)



- **Pour lancer des missiles**
  - ***http://www.linuxdevices.com/news/NS4667725014.html***
- **Dans les télés Sony**
  - ***http://www.sony.net/Products/Linux***
  - ***http://products.sel.sony.com/opensource/source_tv.shtm***
- **Des robots maison ou pro**
  - ***http://www.linuxrobots.org***
  - ***http://www.aldebaran-robotics.com/***
- **Pour traire les vaches**
  - ***http://www.linuxdevices.com/news/NS4275702675.html***
  - ***http://www.linuxdevices.com/news/NS5430405302.html***
- **Pour faire des crèmes glacées**
  - ***http://www.generation-nt.com/linux-open-source-actualite-13082.html***

Source: "Embedded Linux -Petite introduction pour conférence sommaire"
-- PARINUX – Gilles BLANC, palpatine42@gmail.com

Linux (2/4)

- **Caractéristiques**
  - **Faible empreinte mémoire** par rapport aux capacités
  - Efficace**,** optimisé**, performances élevées**
  - Beaucoup d'applications dans **l'espace utilisateur** (support Java notamment)
  - **Modulaire,** paramétrable très finement
  - Enorme support matériel**,** réseau, **portabilité**
  - **Développement *user friendly* (hôte/cible)**

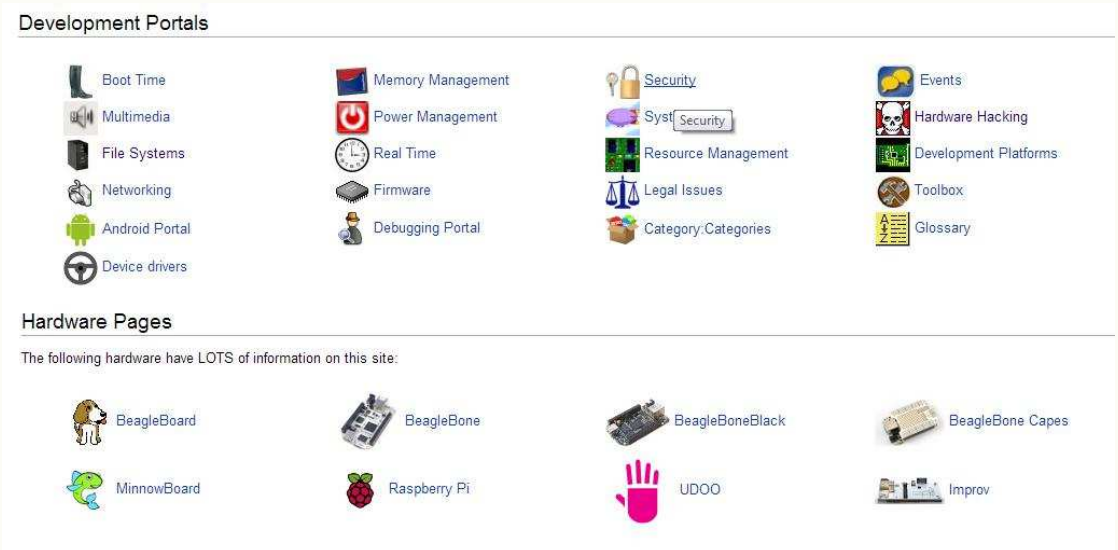
Linux (3/4)

- Solutions propriétaires basées sur Linux
  - Mobilinux (MontaVista Software)
  - LynuxWorks/BlueCat
  - Sysgo/ElinOS,
  - STMicroelectronics/STLinux 2.0
  - Koan Software/KaeilOS
  - …

- Solutions libres
  - Angström Linux
  - Embedded Ubuntu
  - Embedded Debian
  - Embedded Gentoo
  - FREESCO
  - µLinux
  - …

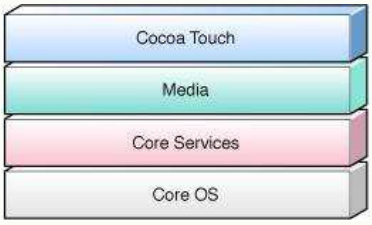
Linux (4/4)

- http://eLinux.org



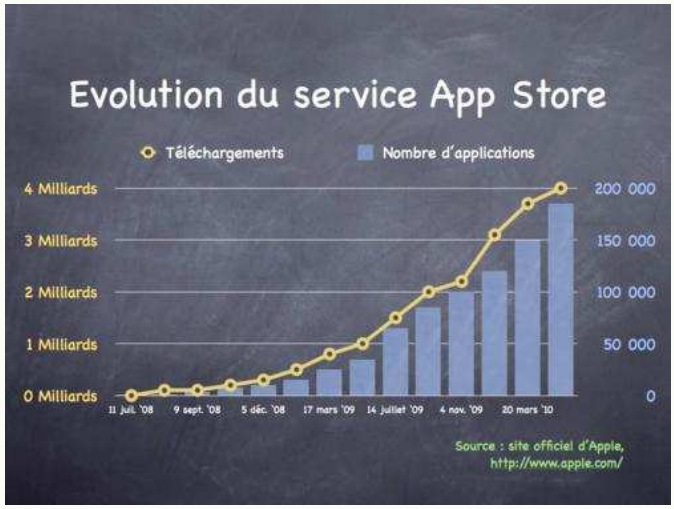
iOS (1/2)

- Noyau hybride (*XNU*)
  - Micro-noyau *Mach*
  - *BSD*
- Multitâche
- 4 couches d'abstraction



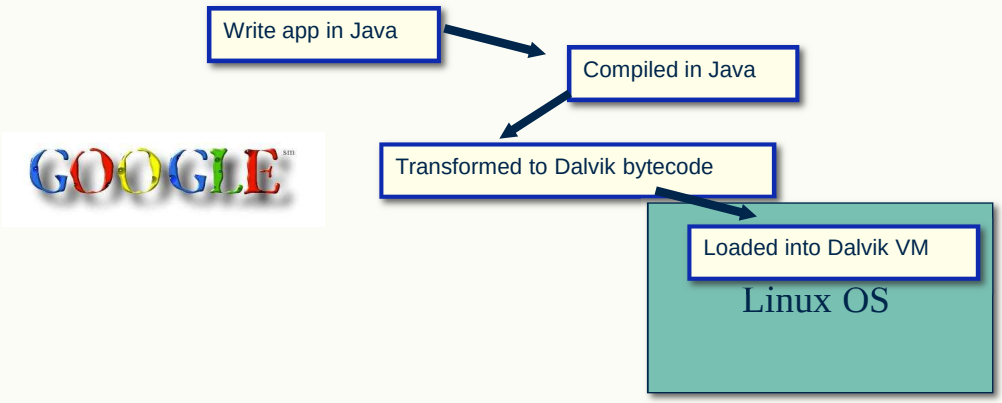
iOS (2/2)

- Evolution sans précédent des applications disponibles sur l'*App Store.*



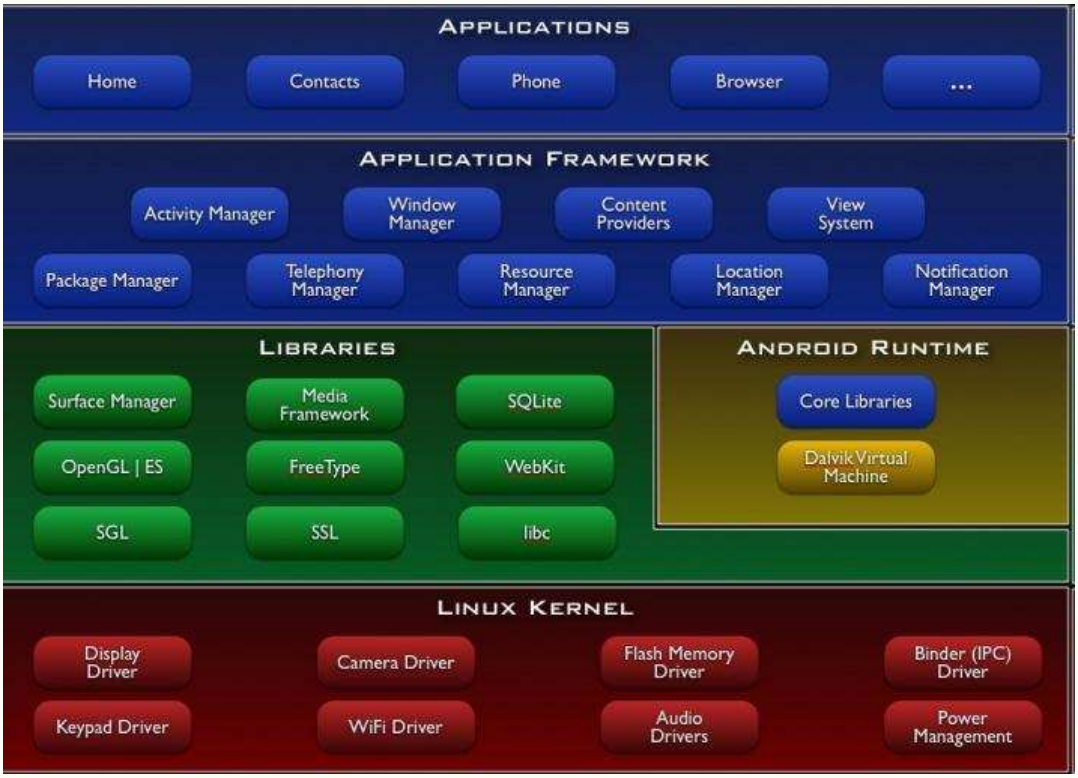
Android OS (1/3)

- Nouvelle approche de développement d'un téléphone ouvert
- http://www.openhandsetalliance.com/



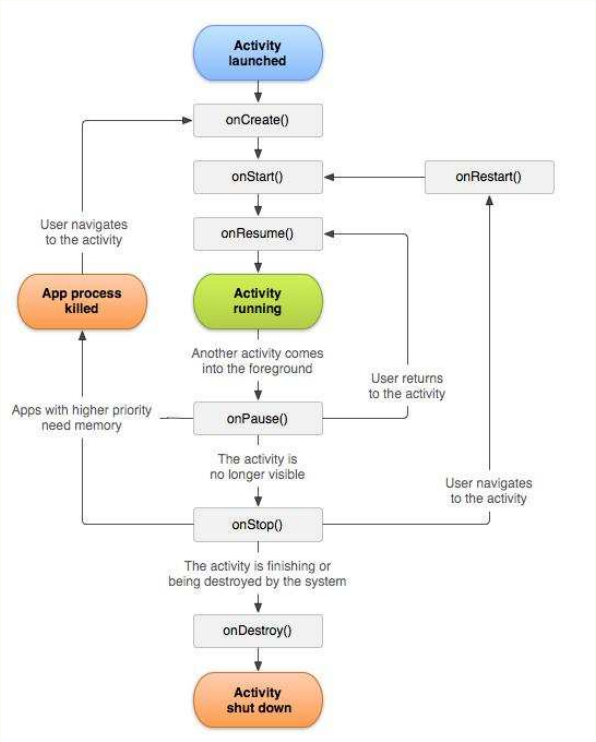
- **Java VM Dalvik**
  - **Développé par les ingénieurs de Google**
  - **Orienté registres (et non piles comme une JVM classique)**

Android OS (2/3)



Android OS (3/3)

- **Application**
  - ***Activity***
- **OOM**
  - ***Out-of-Memory killer***



Autres OS embarqués

- **Téléphones portables**
  - **iPhone OS (basé sur OS X)**
  - **Palm OS**
  - **Openmoko Linux**
  - **Internet Tablet OS**
  - **BlackBerry OS**
  - **…**
- **Routeurs, gateways**
  - **CatOS (Cisco Systems)**
  - **Cisco IOS**
  - **Inferno (Bell Labs)**
  - **IOS-XR**
  - **CyROS**
  - **JunOS (Juniper Networks)**
  - **…**
- **Embarqués**
  - **A/ROSE**
  - **polyBSD (embedded NetBSD)**
  - **ROM-DOS**
  - **MINIX 3**
  - **T2 SDE**
  - **.NET Micro Framework**
  - **OS/RT**
  - **Open AT OS**
  - **…**
- **Music Players**
  - **ipodlinux**
  - **Pixo**
  - **RockBox**
  - **…**

RTOS

- Caractéristiques d'un RTOS
  - Permet le support d'applications temps-réel, donc d'applications nécessitant le déterminisme (temporel/logique) et la fiabilité
  - Simplicité du noyau (taille ~10Ko)
  - Ordonnancement robuste des tâches
  - Objets de communications et de synchronisation prévus pour de telles applications
    - Timeout
    - Latence minimale (interruptions & ordonnancement)

Standards dans les RTOS

- Les RTOS peuvent être conformes à des standards rigoureux
- La norme POSIX est la référence en la matière
  - POSIX est issue du monde UNIX.
  - La norme *POSIX 1003.1b* couvre les aspects standardisation des interfaces et des services pour les applications temps-réel. Il y en a d'autres (1003.1d, 1003.1j), mais ne sont pas utilisées.

- D'autres standards
  - OSEK/VDX (Automobile)
  - ARINC 653 (Aéronautique)
  - Java RTS
  - ...

Posix 1003.1b

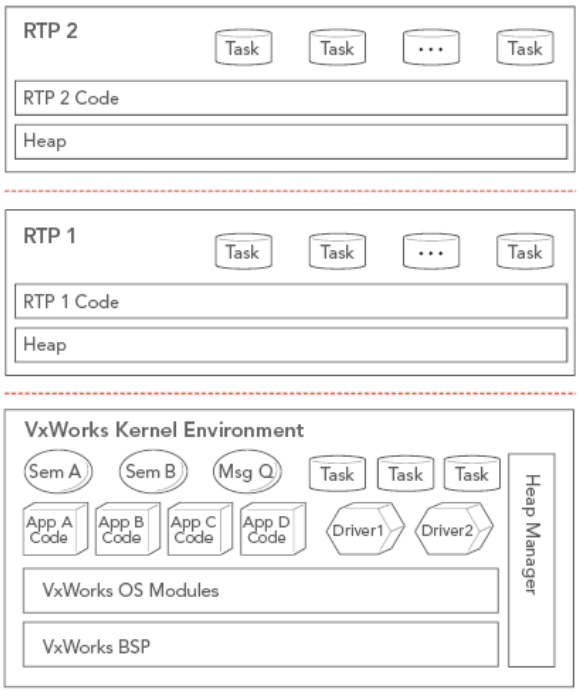
- Les fonctionnalités suivantes sont spécifiées dans POSIX 1003.1b:
  - Timers: periodic timers, delivery is accomplished using POSIX signals
  - Priority scheduling: fixed priority preemptive scheduling with a minimum of 32 priority levels
  - Real-time signals: additional signals with multiple levels of priority
  - Semaphores: named and memory counting semaphores
  - Memory queues: message passing using named queues
  - Shared memory: named memory regions shared between multiple processes
  - Memory locking: functions to prevent virtual memory swapping of physical memory pages

Exemples de noyaux RTOS

- RTOS (noyau/micronoyau)
  - VxWorks (WindRiver)
  - WinCE
  - QNX
  - RTOS-32
  - OS-9
  - ThreadX
  - LinuxWorks
  - PikeOS (SysGO)
  - iRMX (tenAsys)
  - RTEMS
  - FreeRTOS
  - (OpenQNX)
- Cohabitation de domaines RTOS et non-RTOS
  - RTX
  - VxWin
  - RTAI
  - Xenomai
  - …

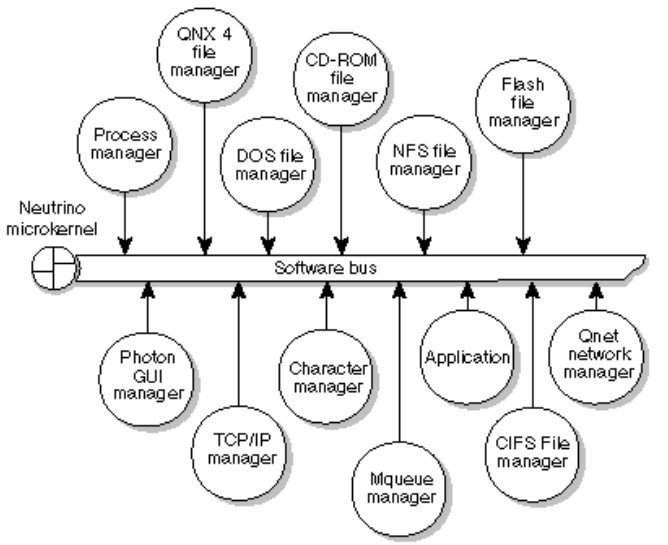
VxWorks

- Protection des espaces d'adressage grâce à l'utilisation de la MMU
  - RTP: Real-Time Process
  - Présence espace user/kernel
- Noyau
  - Ordonnancement préemptif + RR
  - 256 niveaux de priorité
  - Nombre illimité de tâches
- Tâches périodiques gérées par une routine associée à un *timer* périodique et des sémaphores binaires
- IDE basé sur IBM/Eclipse (Java)
- Utilisé par la NASA et beaucoup d'industries (domaine spatial, médical, automation)



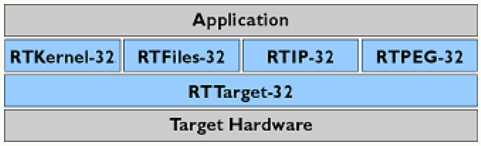
QNX

- Micronoyau (très peu de fonctionnalités dans l'espace noyau, un maximum dans l'espace utilisateur)
  - Micronoyau: CPU scheduling, interprocess communication, interrupt redirection and timers
  - Si un pilote échoue, ou n'importe quelle autre application, le noyau est préservé.
- Ultra-rapide dans la commutation de tâches
- Contient un framework graphique (Photon microGUI)
- Très bon support pour le multicore (SMP)
- Devenu *open source* depuis fin 2007
  - http://www.openqnx.com



RTOS-32

- Basé sur un sous-ensemble de l'API Win32 de Windows
- Noyau
  - Empreinte mémoire de 16 Ko
  - Protection mémoire via MMU
- Peut charger des .DLL de Windows
- Pas de royalties
- IDE de type Borland, Visual Studio

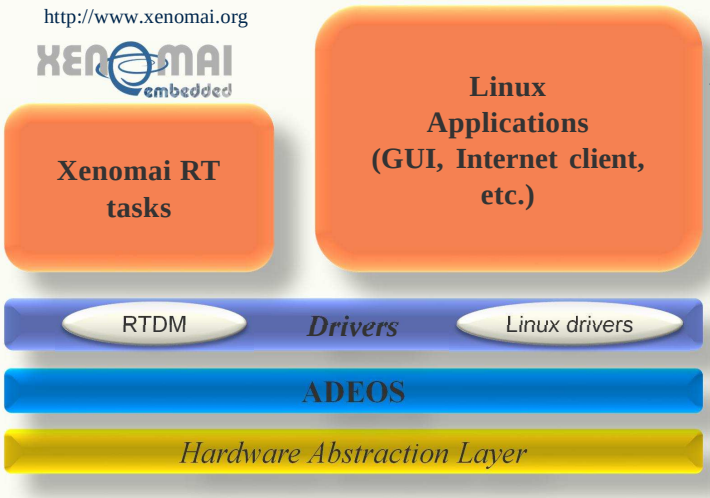


RTTarget-32	RTKernel-32	RTFiles-32	RTIP-32	RPEG-32
-Bootstrap -Librairie de runtime -Drivers -Compression -Sys. fichier RAM	-Ordonnanceur -Sémaphores -Boîtes aux lettres -Communication synchrone	-Systèmes de fichiers disque dur/disquette -Allocation contigüe -Requêtes multiples	-Communication -Stack TCP/IP	-Portable Embedded GUI

Co-habitation OS/RTOS

- Xenomai/Linux
- RTX (Windows)
- RTAI
- …

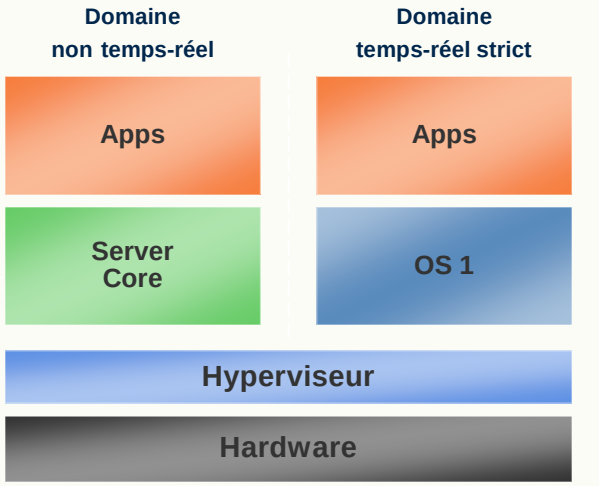
http://www.xenomai.org



Les applications peuvent envoyer/recevoir des messages/events en provenance du domaine RT

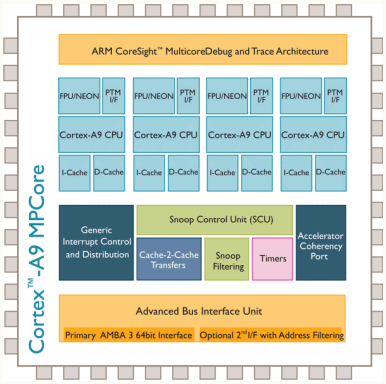
Virtualisation (1/2)

- La virtualisation dans l'embarqué offre de nombreuses perspectives
  - Sécurité par l'isolation des domaines d'exécution
  - Hétérogénéité des OS
  - Prototypage / déploiement
  - *Live Migration*

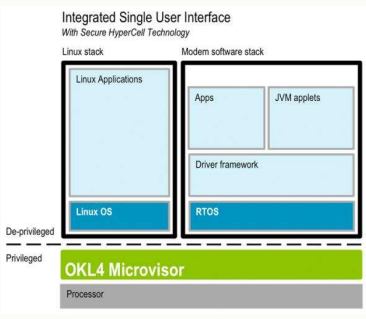


Virtualisation (2/2)

- Utilisation optimale des architectures *multi-core*
  - Cortex-A9 (quad-core)



- Différentes approches
  - Monolithique
    - EmbeddedXEN (http://sourceforge.net/projects/embeddedxen)
    - XEN
    - KVM (Linux)
  - **Micro-noyau**
    - **OKL4**, **CodeZero**, eVM (TenASys), etc.
    - http://www.ok-labs.com
    - http://www.l4dev.org



Aspects logiciels dans l'embarqué

- Le *software* dans les systèmes embarqués
- Systèmes d'exploitation pour l'embarqué
- **Aspects liés au design de logiciel embarqué**

Besoins en modélisation

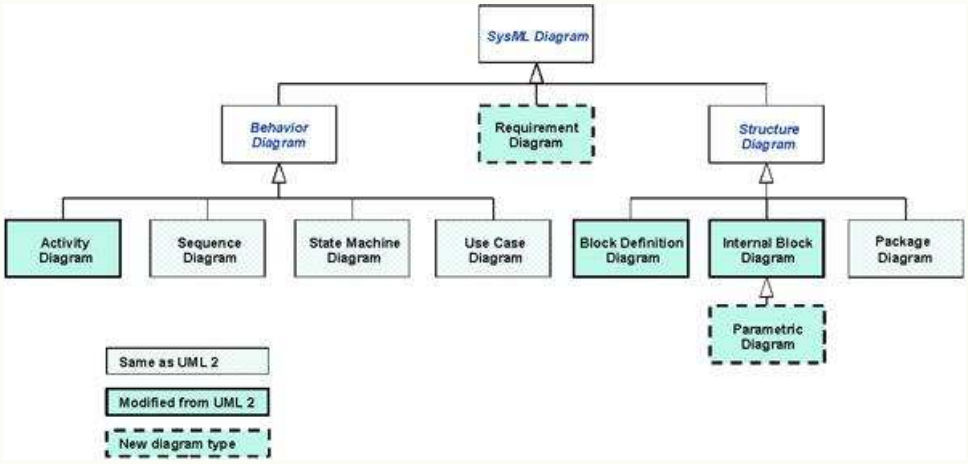
- **Le développement de systèmes embarqués nécessite des compétences pluridisciplinaires.**
  - **Besoin de communiquer clairement les spécifications du système**
  - **Besoin d'exprimer clairement les contraintes (fonctionnelles, temporelles, environnementales) du système**
- **Le coût de développement peut être élevé.**
  - **Besoin d'anticiper les problèmes et d'appréhender les limites du système**
- **Les risques de dysfonctionnement peuvent entraîner des conséquences catastrophiques.**
  - **Besoin d'assurer le comportement du système**

Modélisation formelle

- Langages de description de matériel
  - Supportent les concepts spécifiques matériels
  - VHDL, Verilog
- Langages de description des architectures
  - Définition formelle de l'architecture de haut niveau d'un système complexe
  - SystemC ,SpecC, SysML
  - SysML
    - http://www.sysml.org/
- Langages orientés objet
  - Description des systèmes complexes à un très haut niveau d'abstraction par sa décomposition en sous-systèmes
  - UML, Realtime UML
- Langages pour la modélisation des systèmes temps réel
  - Approches synchrones: Lustre, Esterel, Signal
  - Approches asynchrones: SDL, Realtime UML

SysML (OMG/UML)

- Modélisation niveau système basée sur UML
- Combinaison software/hardware
- SysML
  - http://www.sysml.org

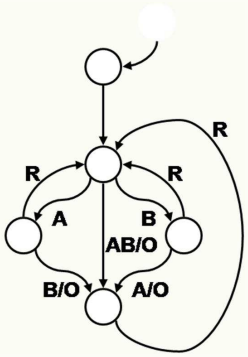


Esterel

- Un programme Esterel est un automate hiérarchique et parallèle qui attend des signaux, réagit à un stimulus composé de signaux en changeant d'état et en produisant de nouveaux signaux.
  - Notion de base: signal
    - Un signal est présent/absent.
    - Hypothèse : temps de réaction nul !
    - Un signal émis ou reçu est visible pendant toute la réaction du programme

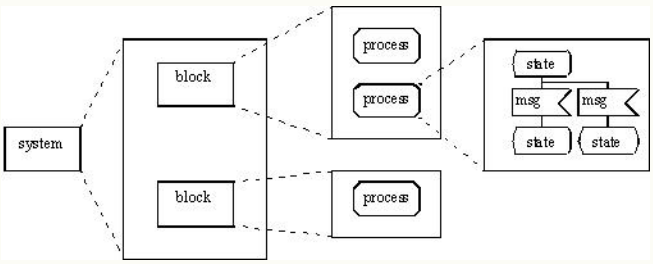
- Origine : INRIA (Sophia Antipolis)
- Distribution commerciale :
  - Esterel Studio - Esterel Technologie
- Utilisations industrielles :
  - Dassault Aviation

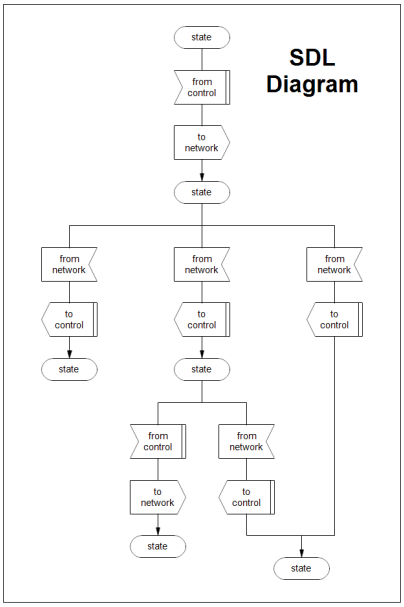
```
loop
  [
        await A
        ||
        await B
  ];
  emit O
each R
```



SDL

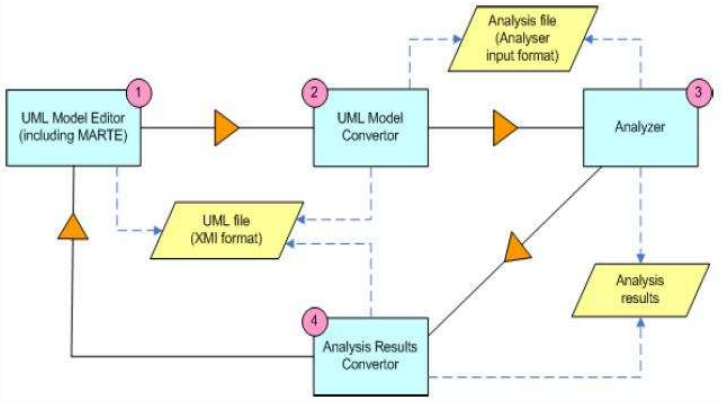
- *Specification and Description Language*
- Ce langage est issu du domaine des télécommunications (ITU-T)
  - Description de processus asynchrones
  - Spécification de protocoles
  - Difficile d'exprimer les contraintes temporelles
- Notion de système, blocs et processus
  - Communication par signaux via des canaux





UML

- Langage de modélisation d'applications orienté-objet (OO)
  - Dimensions fonctionnelles, statiques, dynamiques
  - Différentes classes de diagrammes
- La modélisation temps-réel nécessite une extension d'UML
  - UML-RT, RT-UML, **MARTE**, ACCORD
- Extension de la notation pour l'expression de contraintes temporelles
- Environnements logiciels
  - Rational Rose (UML-RT)
  - Rhapsody (RT-UML)
  - Papyrus
    - http://www.papyrusuml.org

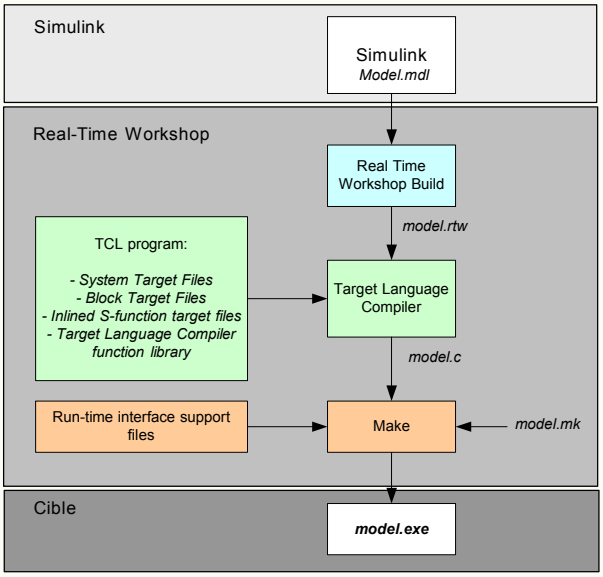


Rapid Prototyping

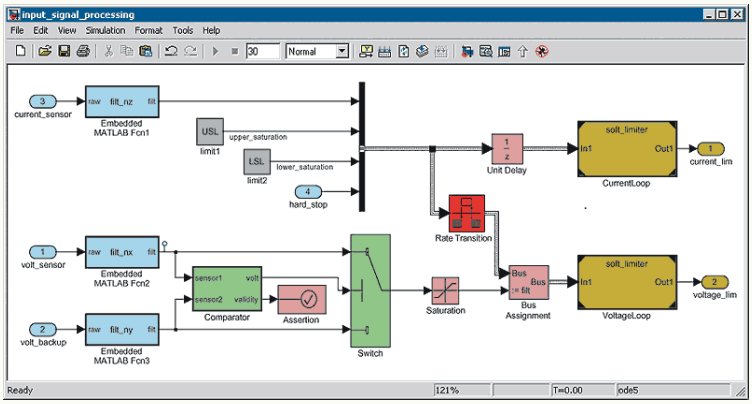
- Développement d'une application au moyen d'un environnement intégré complet.
  - Modélisation au travers d'outils graphiques
  - Simulation du système
  - Génération de code automatique pour des architectures cibles
  - Déploiement et mise au point

Simulink & Realtime Workshop

- **Modélisation et simulation de systèmes dynamiques embarqués**
  - **Temps discret et continu**
  - **Algorithmes**
  - **Fonctions complexes dédiées au traitement des signaux**
- **Vaste choix de composants graphiques**
- **Génération automatique de code pour des cibles RTOS génériques (ou spécifiques)**



Exemple




```
…

/* Subrate 2 task */
void tSubRate_2(void * cookie)
{
  int status;
  unsigned long overrun;
  while (1) {
    /* Set model inputs associated to subrate here */
    slbenchmodel_step2();
    status = rt_task_wait_period(&overrun);
    if (status)
      printf ("Error rt_task_wait_period !\n");
    if (overrun) {
      printf("Rate for SubRate 2 too fast.\n");
      cleanExit(0);
    }

    /* Write model outputs associated to subrate here
*/
  }
}

/* Subrate 3 task */
void tSubRate_3(void * cookie)
{
  int status;
  unsigned long overrun;
  while (1) {
    /* Set model inputs associated to subrate here */
    slbenchmodel_step3();
    status = rt_task_wait_period(&overrun);
    if (status)
      printf ("Error rt_task_wait_period !\n");
    if (overrun) {
      printf("Rate for SubRate 3 too fast.\n");
      cleanExit(0);
    }

    /* Write model outputs associated to subrate here
*/
  }
}

…
```