M.DESCHAMPS Mathieu M.BIGNAUX Ronan

Tech-stories : Aspects des projets FLOSS R&D.

Table des matières

Le processus R&D

R&D FLOSS

Introduction

Cartographie

Marché : plus 4 Milliards €

Acteurs : SSII & éditeurs de logiciels. Presque 50% existent depuis plus de 15 ans

Secteur : environ 70% pour la défense, le militaire, l'espace et l'automobile

CA : 90% réalisé en France.

Invest. R&D embarquée : 7% du CA

Activité embarqué minoritaire : pour environ 80% des Acteurs

Modèle économique : 44% Mixtes (4% de "puristes")

Pérennisation technologie embarqué : 70% privateur (14% OpenSource)

57% Source Closing

13% Matériels IP ""standard""

Qu'est-ce que la Recherche Développement ?

"Systematic activity combining both basic and applied research, and aimed at discovering solutions to problems or creating new goods and knowledge."

Un processus au long cours composé d'activités combinant recherche fondamentale et opérationnelle dans le but de concevoir, de réaliser et d'améliorer un produit/service au moyen d'une solution innovante.

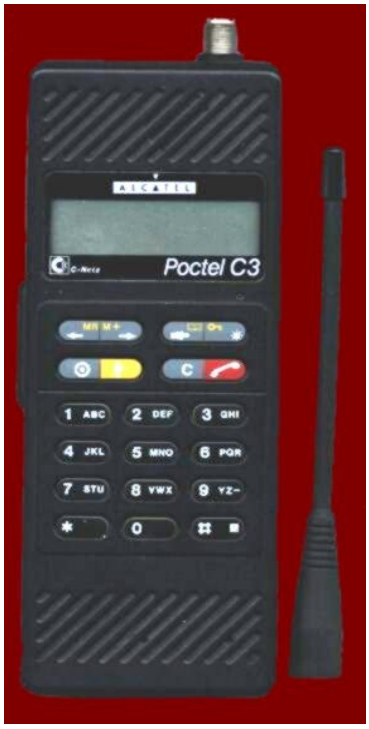
Commentaires

Source

www.businessdictionary.com

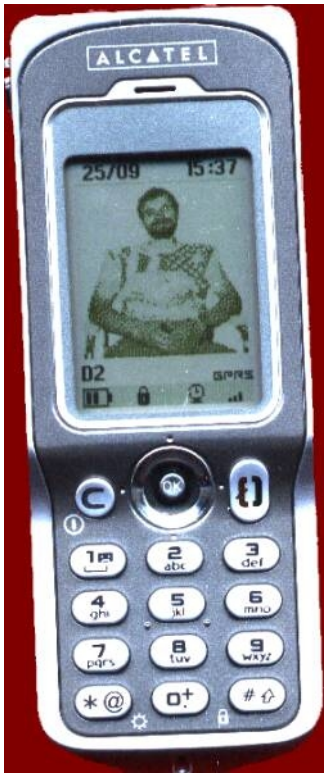
20 ans de RD Télécom : le téléphone portable Alcatel

S'il est né au début du siècle dernier, le téléphone n'est devenu (trans)portable qu'au début des années 1990.


ALCATEL
Poctel C3
MR M+
C
1 ABC
2 DEF
3 GHI
4 JKL
5 MNO
6 PQR
7 STU
8 VWX
9 YZ-
0


ALCATEL
NO
OK
ONE TOUCH
Easy

Alcatel One Touch Easy Bi-Bande



OV T770

Crédits Photos : Copyright 2008 - Handy-Sammler.de

Commentaires

L'Alcatel SEM 335 M/TC

1988

3,77kg

Poctel C3

1990

585g

35 cm x 8 cm x 3,5 cm

OTE

1997

200g

15 cm x 6 cm x 4 cm

(passage à la carte à puce SIM)

OTE

1999

200g

15 cm x 5 cm x 3cm

Dualband 900/1800MHz

OTE 715

2002

88g

11 cm x 4 cm x 2 cm

Alcatel OT-V770

2008

75g

10 cm x 5 cm x 1 cm

Crédits Photos (danke Elmi)

Pourquoi démarrer une activité R&D ?

Objectifs

Qualifier/Quantifier une faisabilité : R&D "exploratoire", la recherche d'opportunités nouvelles

Maintenir une position sur le marché : R&D "exploitation", la recherche depuis l'existant sur objectifs

Valider une démarche, certifier un procédé : R&D "consolidation", la recherche servant de R&D étalon

Durée : une précision.

Une R&D ne dure pas le temps de remplir un objectif...

elle dure le temps de l'épuiser.

Et à l'échéance on estime si l'objectif est rempli.

Qu'est-ce que la R&D des Logiciels Libres et Open Sources (FLOSS) ?

La même chose avec les bons logiciels ? ;)

Pas seulement...

- Les outils,
- Les procédés,
- Les acteurs,
- L'organisation de projet

... diffèrent.

La R&D FLOSS s'épanouit grâce à l'effort conjoint des Logiciels Libres et Open Sources sur deux fronts dans :

- les projets fondateurs pour l'ouverture et
- les entreprises commerciales pour l'extension

Deux années de RD FLOSS sur le projet OpenMoko

FIC Neo 1973

La R&D liée à OpenMoko donne naissance un produit innovant exploité par la société FIC.

Pour conclure

C'est un continuum qui se découpe en phases, en projets indépendant (en principe !) du cycle de vie d'un produit.

- But : donner naissance à un produit/service issu de la plus-value apportée à l'état de l'art.
- Constat : c'est l'effort commun des initiatives privées et publiques qui permet son essor.

La R&D FLOSS a donc un caractère essentiel et cyclique, synergique et indépendant.

Organisation

Fonctionnement

Constats

L'utopie des spécifications immuables perdure.

L'erreur du Test-driven développement

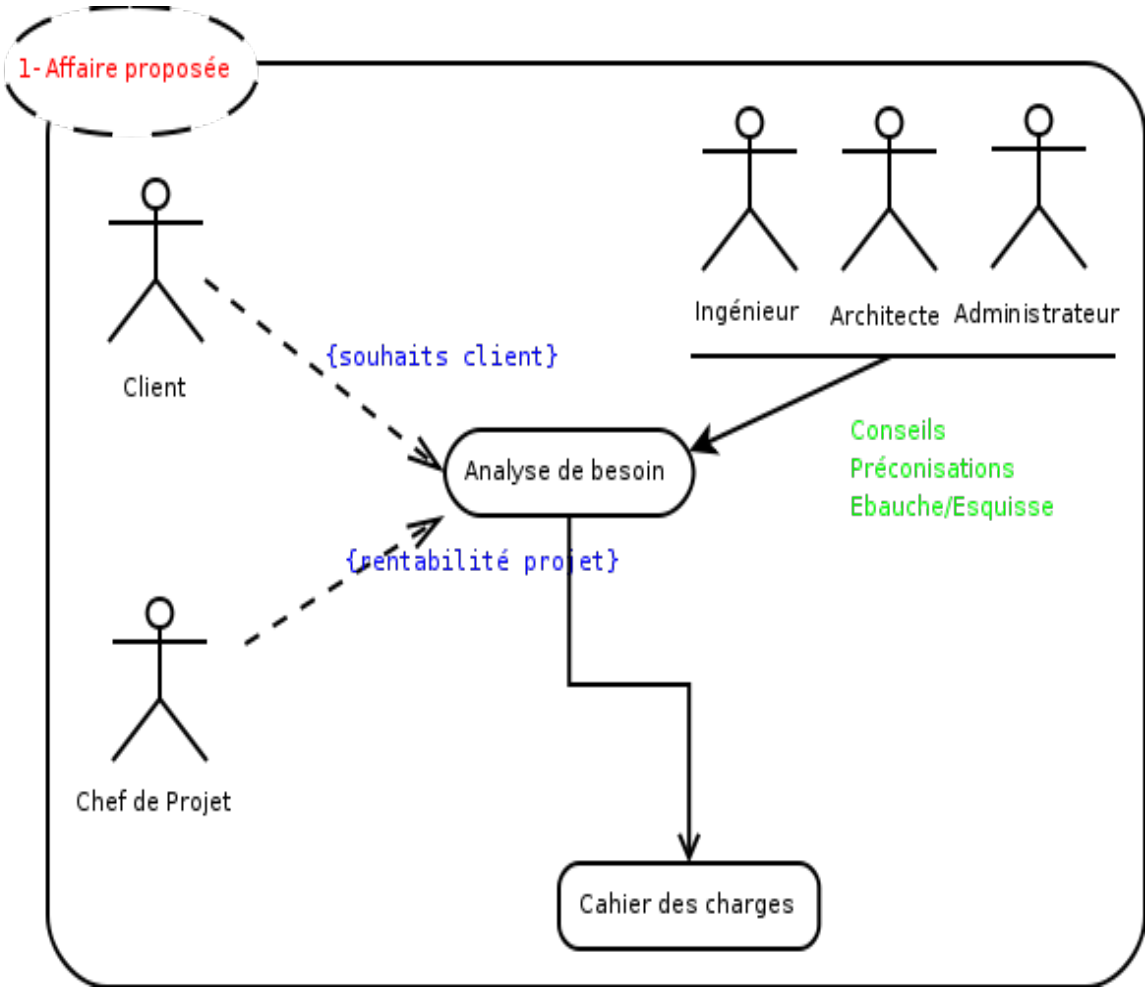
Si les DCU des méthodes UP sont presque inefficace pour la méthodologie en "couche basses" ...

Ils restent applicables pour l'organisationnel :

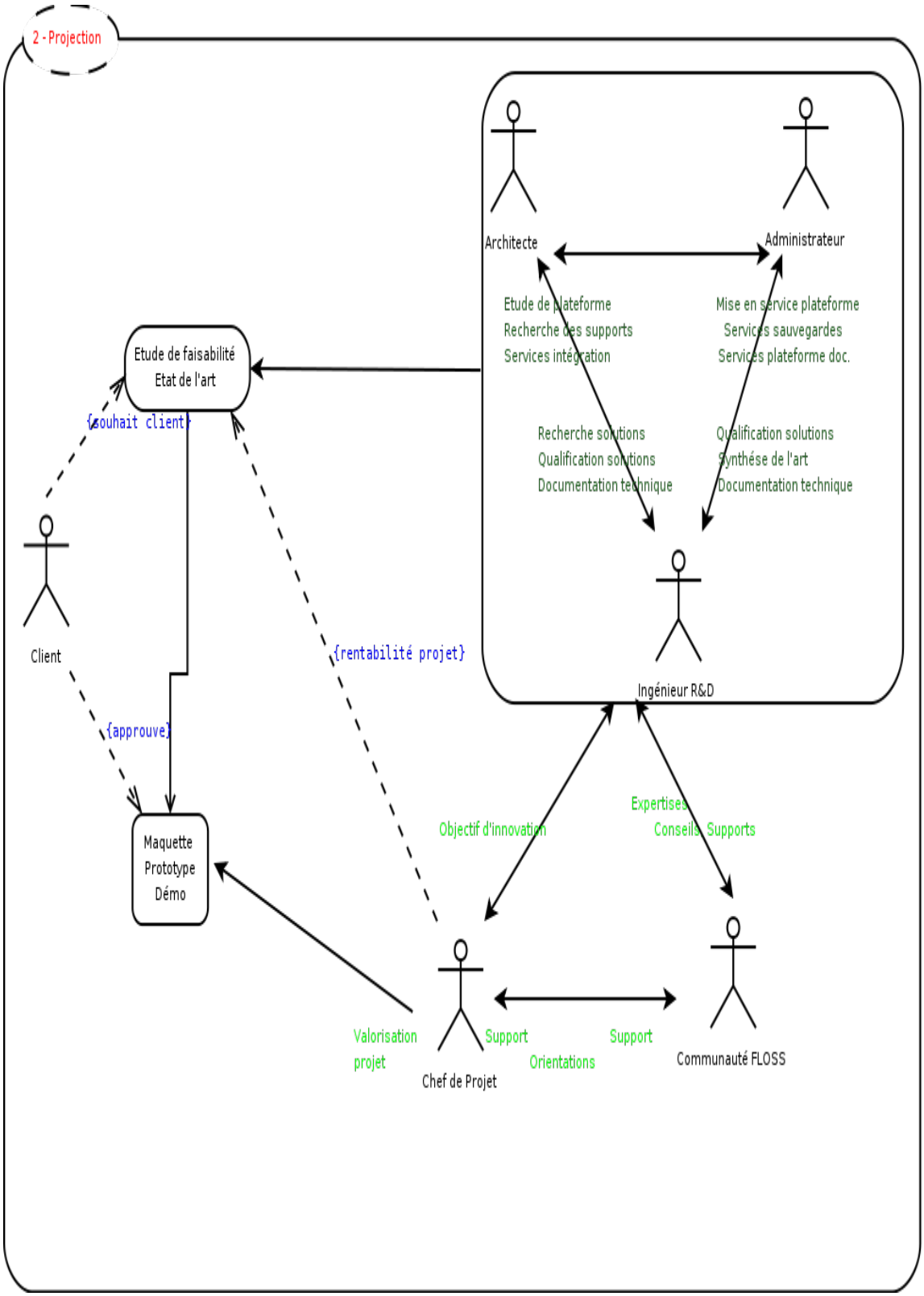
- Qui est celui qui va utiliser mon code/mon service ?
- Comment va-t-il l'utiliser ? Quels sont ses entrées ?
- etc..

Les méthodes agiles gagnent du terrain...

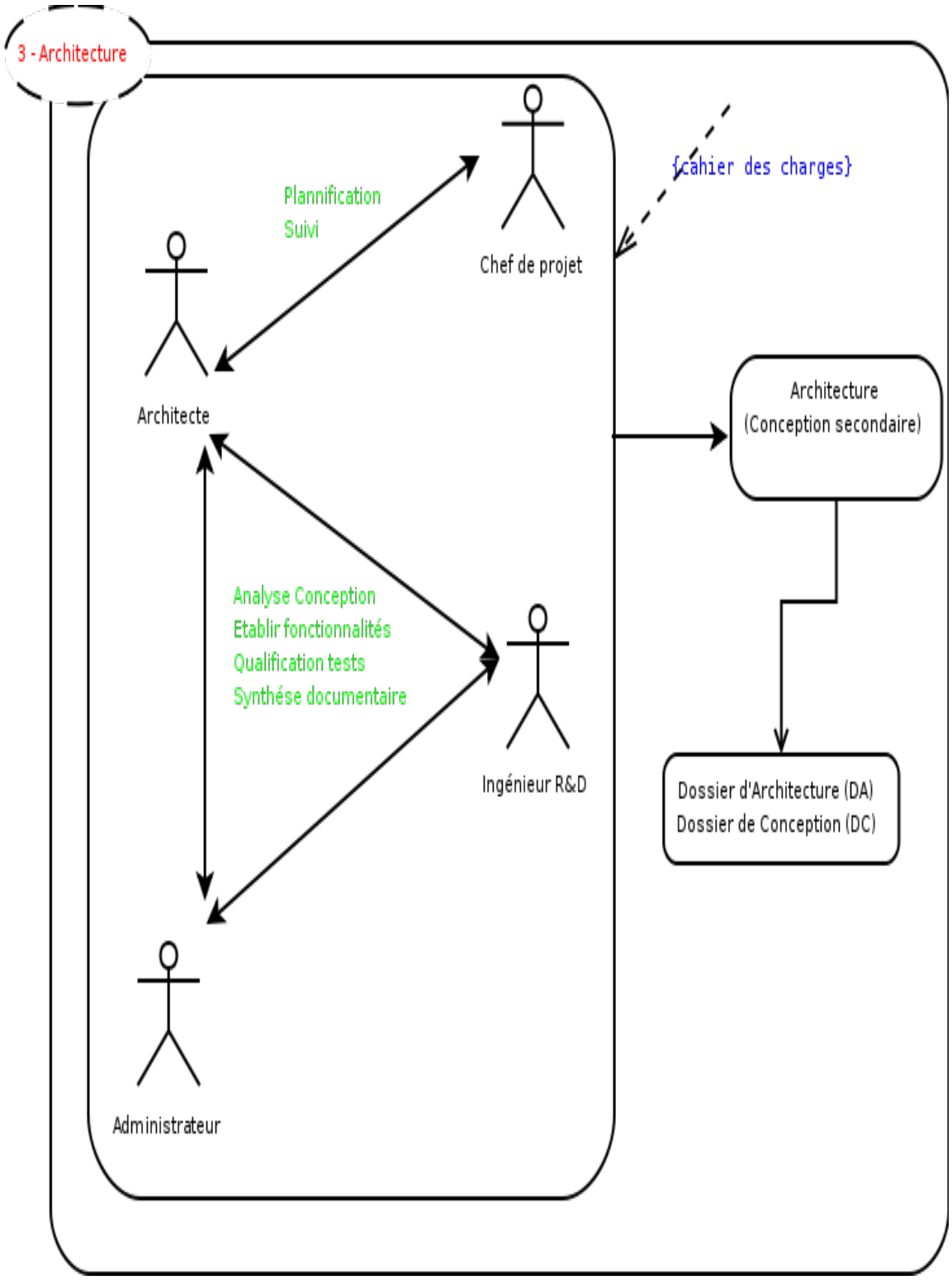
Adapter l'organisation à la R&D FLOSS

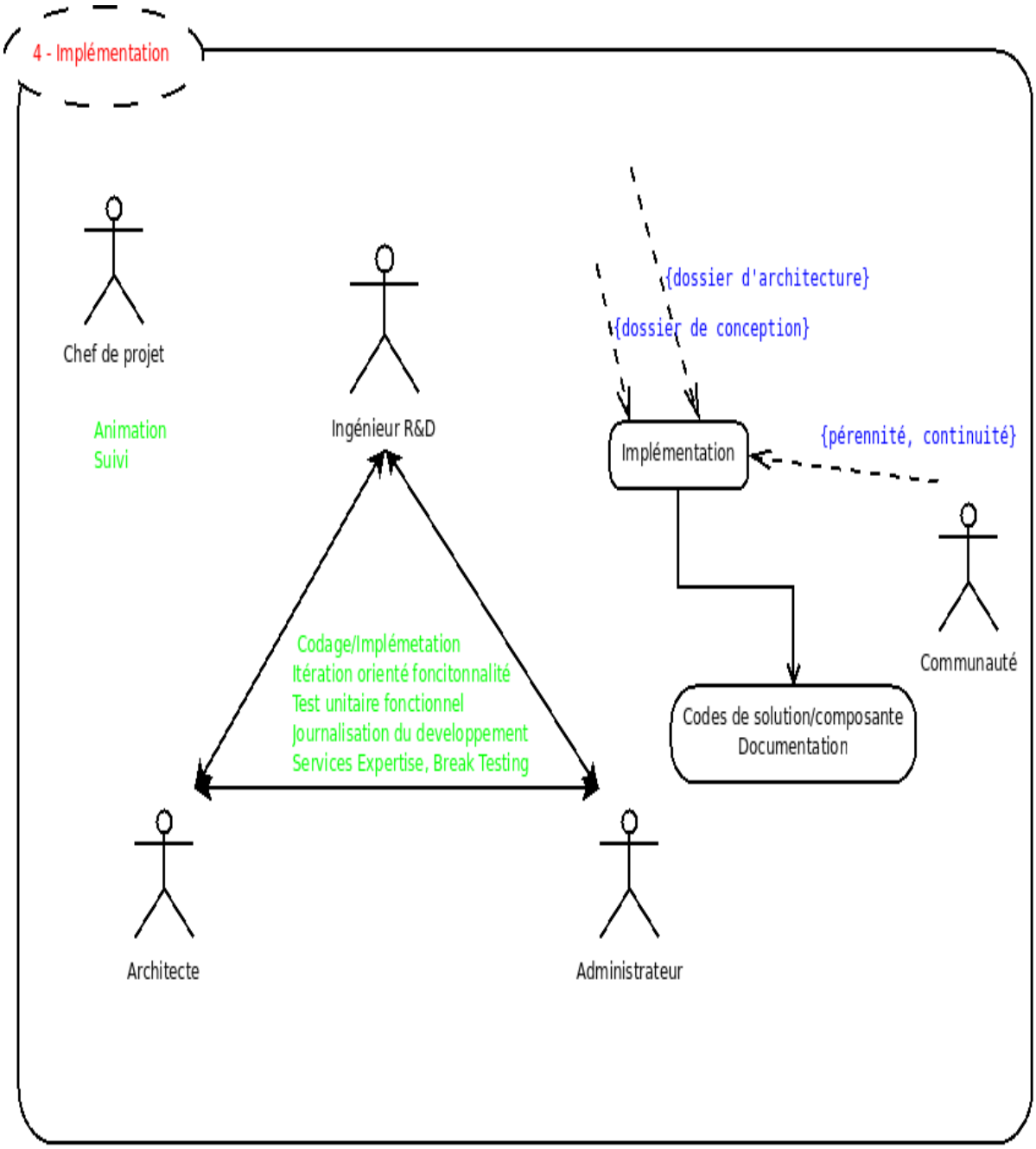


Ecouter-Intervenir-Echanger : Répondre au besoin

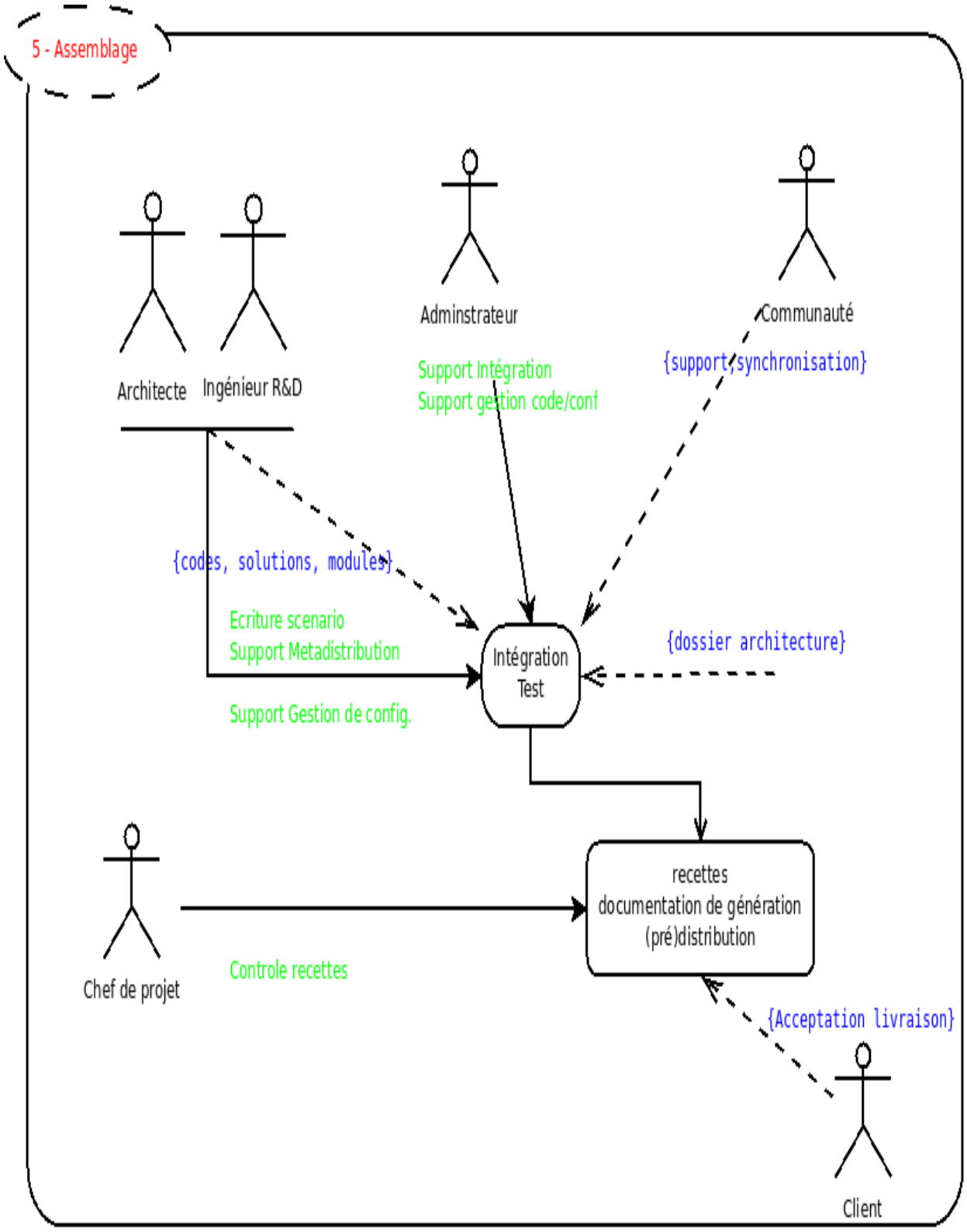


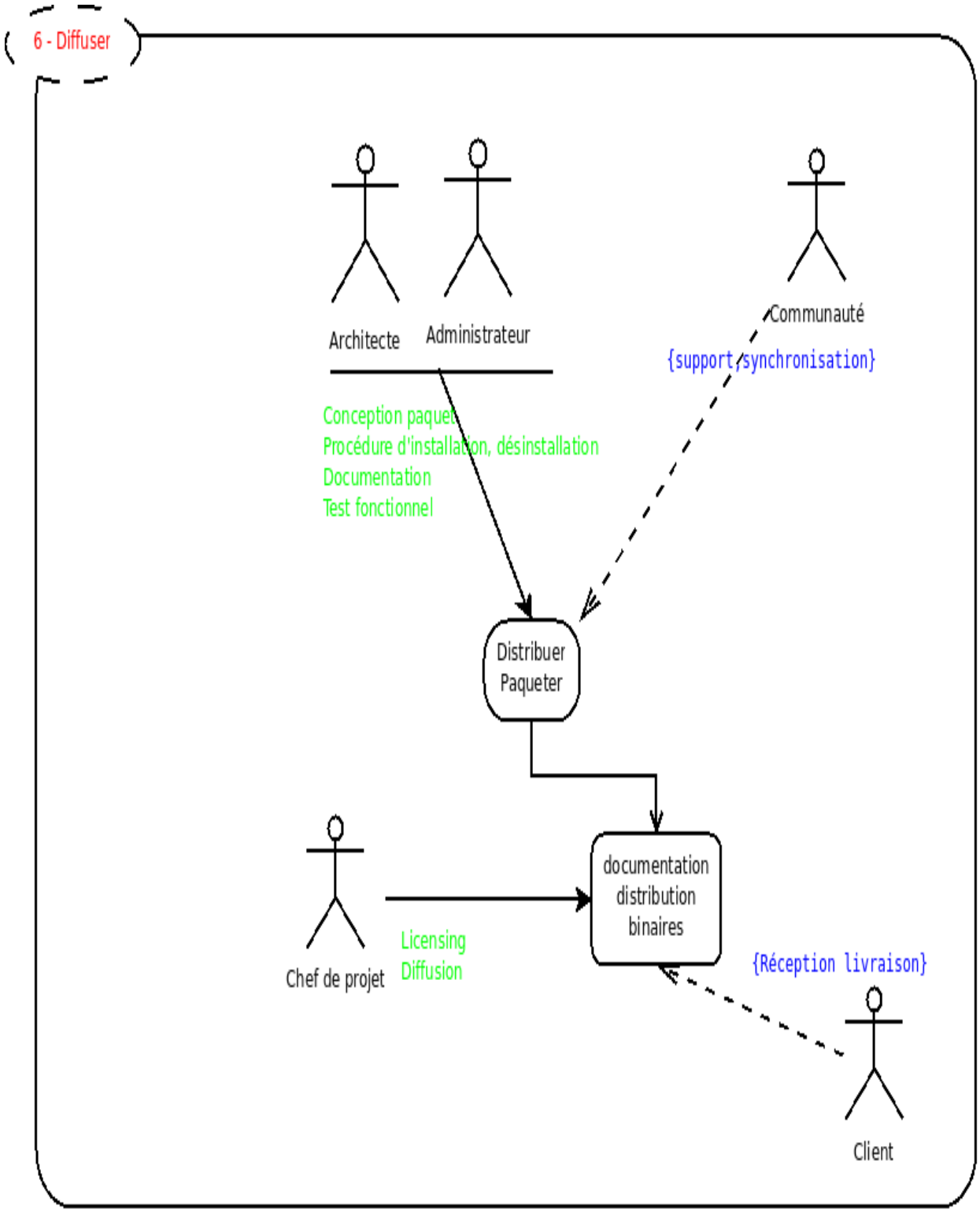
Collecter-Evaluer-Etablir : Estimer une faisabilité


3 - Architecture
Chef de projet
{cahier des charges}
Plannification
Suivi
Architecte
Architecture
(Conception secondaire)
Analyse Conception
Etablir fonctionnalités
Qualification tests
Synthèse documentaire
Ingénieur R&D
Dossier d'Architecture (DA)
Dossier de Conception (DC)
Administrateur



Coder-Tester-Journaliser : Implémenter


5 - Assemblage
Architecte
Ingénieur R&D
Adminstrateur
Communauté
Support Intégration
Support gestion code/conf
{support, synchronisation}
{codes, solutions, modules}
Ecriture scenario
Support Metadistribution
Support Gestion de config.
Intégration
Test
{dossier architecture}
Chef de projet
Controle recettes
recettes
documentation de génération
(pré)distribution
{Acceptation livraison}
Client



Licencier-Paqueter-Diffuser : Concevoir une distribution

Elaboration

Associer le maître d'œuvre (si ce n'est pas le rôle du client : associer le client !),

les intervenants de la gestion de projet

aux équipes de développement pour les phases stratégique du projet !

Communication imagée

Simplicité & approche naïve

Découper (sans hacher ou à contrario sans effet tunnel) en sous-projets de quelques mois la R&D logicielle et matérielle

Commentaires

Support logiciel intégré (drivers) pour ce fameux composant bon marché (surcoût du dev. au lieu d'intégrer)

Distribution

La diffusion de version prototype même semi validée, jalonne.

Intégrer en continu

mon(Mythe) : le code e(s)t la documentation

ma(Doxa) : la documentation (n')est utile (qu')au client.

(my)Truth : documenting is('nt) evil !

Organisation de travail

Maintenir un rythme constant de travail

Veiller à la consistance de ce travail

Présence physique, n-nômes de travail

Pilotage ou Y a-t-il un pilote ?

Tracker ou pas ? Chef de projet alors ?

Euh ... <grosse_bêtise> le client ? </grosse_bêtise>

Réponse(s) ?

un des développeurs.

ou un autre.

Pour conclure

1.Transposer les modèles transposables sinon adaptons-les !

2.Elaborer sans clanisme, naïvement. Raffiner utilement et constamment. Aimer trouver certes ! mais chercher

3.Intégrer avec constance. Distribuer sans frénésie. Toujours documenter (par étapes).

4.Equipe à organisation en n-nômes et non par "métier". Responsabilisation & Implication collective, Emulation & Synergie (RIES).

5.**Evoluer au delà du pilotage TopDown**

Outils R&D

Communication

- courriel
- liste de diffusion
- wiki,cms
- forum, groupware

Commentaires

Une myriade d'outils pouvant booster ou plomber un développement.

Non au tout mail. Chercher des moyens adaptés aux contenus.

Outils de développement et d'intégration

- éditeur de code , IDE
- outils de debogguage
- émulation, bac à sable
- système de construction
- logiciels de gestion de versions
- metabuild system / meta distribution

Commentaires

Complexité intégration du code ou Build system : autotools , cmake , scons

SCM : cvs/svn sucks , git/arch rocks

Qualité

- système de suivi de bogues
- tests (unitaires et de non-régressions)
- logiciel d'intégration continue

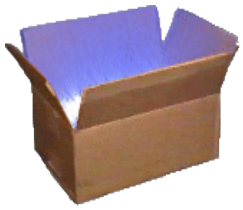
Commentaires

bug tracking system : bugzilla, mantis , trac (outil composite) selon taille d'équipe

Intégration continue = ! test driven

Exemple

OpenEmbedded



Buildroot

Pour conclure

La faisabilité d'une R&D FLOSS c'est **prévoir** et **estimer** l'intérêt/pertinence de ces outils.

L'oublier engendre souvent des retards, des "migrations" et une contre productivité croissante au cours du projet.

Ingénierie pour embarqué

BSP

Le BSP, un point crucial du développement embarqué

Définition

Le BSP représente l'ensemble de la solution :

- BIOS
- Bootloader
- Kernel
- RootFS

Importance de la maîtrise de la chaîne de boot

BIOS : OpenFirmware, Coreboot

Coreboot : BIOS libre

Bootloaders avec moniteur : Das U-boot, RedBoot/ecos, Etherboot

Personnalisation :

- rapidité du boot
- adaptation au média
- tests initiaux
- ergonomie

Coûts :

- économie des composants
- économie en licence

La chaîne compilation

2 chaînes de compilations (croisées, cachées/distantes) :

BIOS/Bootloader et Kernel/Appli.

Glibc ou µClib (et les autres) ?

Librairies auxiliaires...

Hello world ! (dynamically linked, using shared libs, not stripped)

12174 : ELDK

6159 : µClibc

Déboggage ?

Pas de "Stripping"

Flags conseillé :

Pour conclure

Maîtrise, "extensivité", systématique.

Rappelez-vous :

Dimensionner le système de génération de distribution embarqué, c'est encore faire de l'embarqué !

Métadistribution

La distribution

Métadistribution, distribution

La métadistribution fabrique la distribution embarquée.

Utiliser une métadistribution libre permet :

- une indépendance vis à vis d'un constructeur
- une meilleure maîtrise du système
- de capitaliser des compétences et des ressources

Commentaires

Les "métadistributions" des fondeurs fonctionnent mal,
les licences empêchent la réutilisation du code,
ce qui empêche le portage de composantes logicielles libre.

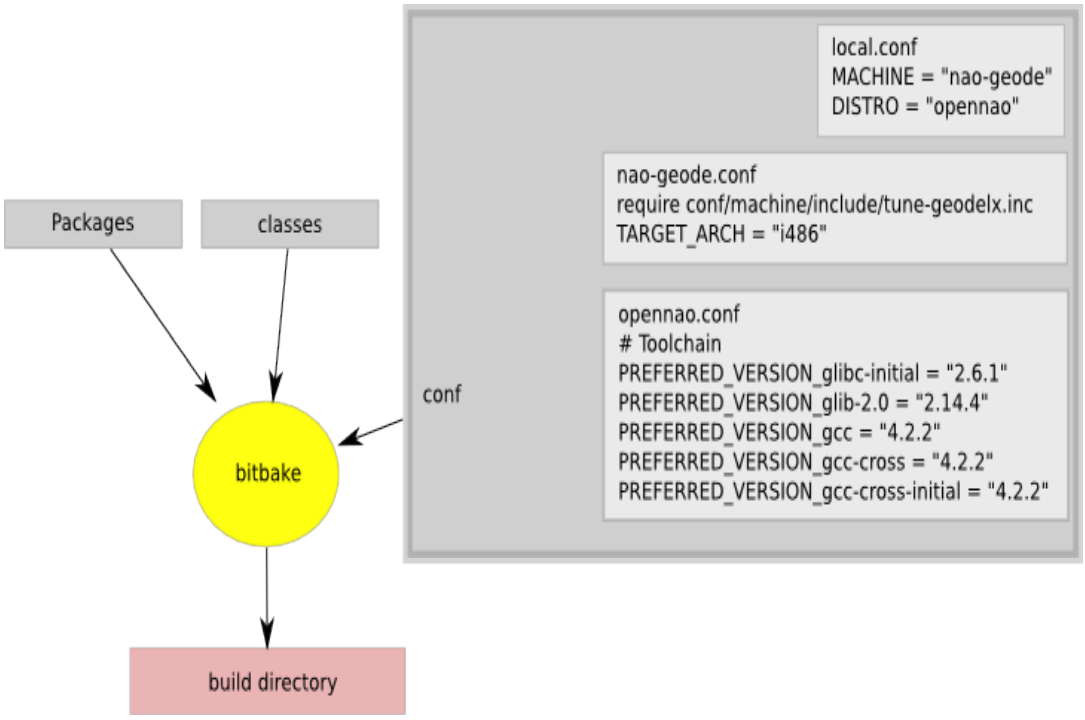


Diagramme simplifié de fonctionnement

OpenEmbedded

Caractère systématique de la génération

Conception pseudo orienté objet : héritage , classe

Tâches adaptées à la création de BSP :

- création de chaînes de compilation croisée
- création d'image du système
- créations de paquets
- gestion d'arbre de paquet

Commentaires

bitbake : task manager. Gère les dépendances de taches et l'ordre de leur exécution

Ecriture des classes en bash & python.

Pour conclure

Pour privilégier le déterminisme et la systématique de la génération :

- fixer le versionnement des paquets,
- centraliser leurs fichiers de configurations,
- conserver les journaux de génération.

Comment dire adieu au longue suites de tests d'intégration !

Embedded Linux

Concevoir son Linux, Fabriquer une image, Débogguer

Comment dimensionner linux ?

- Modulariser
- Démoniser
- Journaliser

Commentaires

maîtriser l'empreinte statique mémoire vive.

Prétexte du module en mode devel : facilité de test (insert/remove), logging kernel, profiling intégré, etc...

Spécifier finement le modules.conf et démoniser dans init.d ou udev (ex : gestion energie)

stripper les modules (en prod)Journaliser et garder des traces (system.map)

Fabriquer son image pour le bootloader

mkimage -A arm -O linux -T kernel -C gzip -a 0x20008000 -e 0x20008000 -n Linux2.6.14 -d zImage uImage-2005-12-15-Thursday

Image Name: Linux2.6.14

Image Type: ARM Linux Kernel Image (gzip compressed) Data

Size: 965280 Bytes = 942.66 kB = 0.92 MB

Load Address: 0x20008000

Entry Point: 0x20008000

Commentaires

Scripts de génération automatique depuis vmlinuz (image)

Connaitre exactement le entrypoint et le loadadress et comprendre pourquoi !

Utile en cas de forensic.

Déboguer un module

Ne pas faire confiance à un kernel qui OOPS.

1. Augmenter la verbosité (make V=1)
2. Utiliser Ksymsoops (/proc/ksyms vs. system.map)
3. Avoir une sonde JTAG/BDI ?

Commentaires

Modulariser le composant qui plante.

Ksymsoops : Prendre la carte de symbole du noyau et... GDB

Le moniteur de la sonde est indépendant/peu intrusif.

KGDB support & BDI support : mutually exclusive !

Mur de l'activation de la MMU : deboggage avant & après, rien à voir !

Modutils et sécurité

Modules owned by root : refusé ou pas ?

Commentaires

Si besoin, configurez Modutils pour qu'il refuse les modules n'appartenant pas à root.

Pensez à vérifier la phase de depmod du kernel aussi (pas de "-r" qui les accepte.)

Dimensionner linux

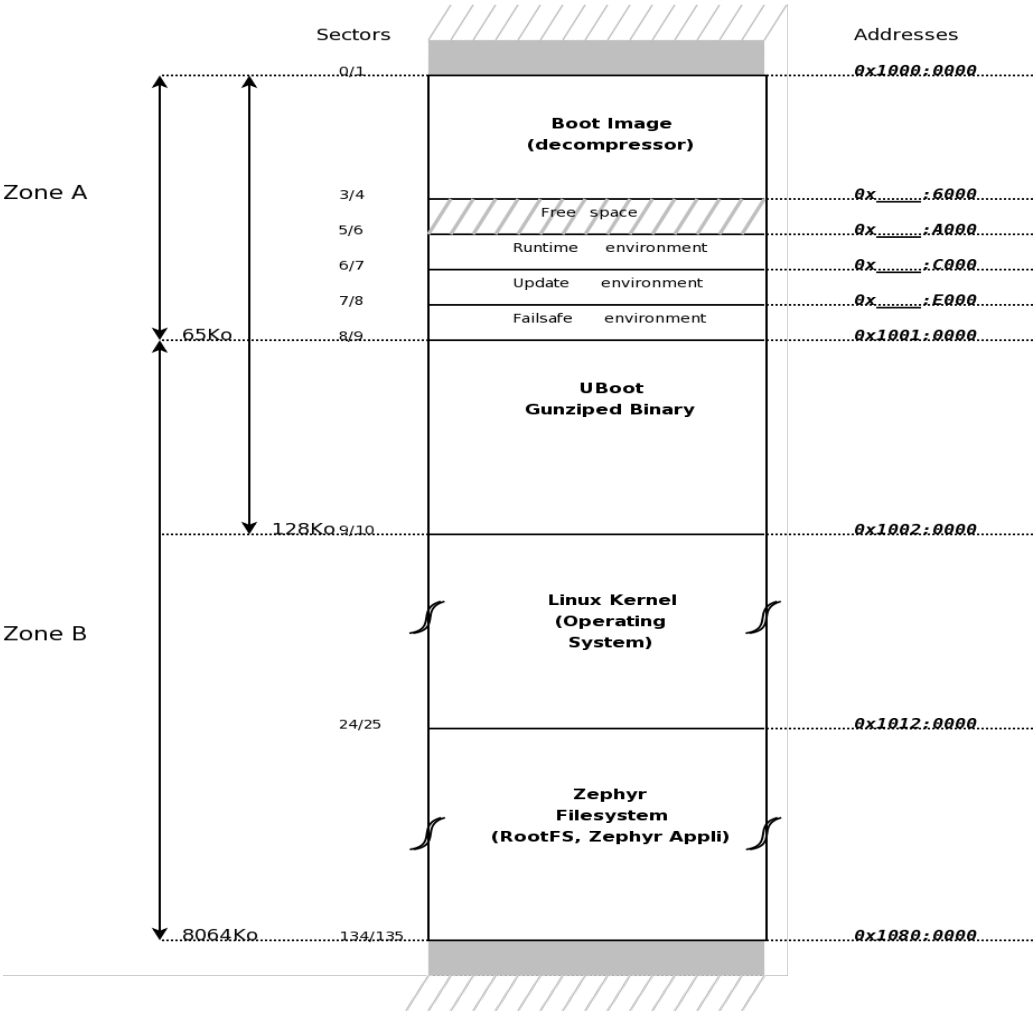


Diagramme mémoire Flash

Pour conclure

Passer un peu (plus, beaucoup, ... ;) de temps pour embarquement le kernel,

pour hacker les interfaces, y ajouter des supports matériels, enlever des branches "mortes" :

Dédier.

Questions ? Animaux poilus ?

Contact

Deschamps Mathieu (mathdesc)
mathdesc@scourge.fr
http://www.scourge.biz