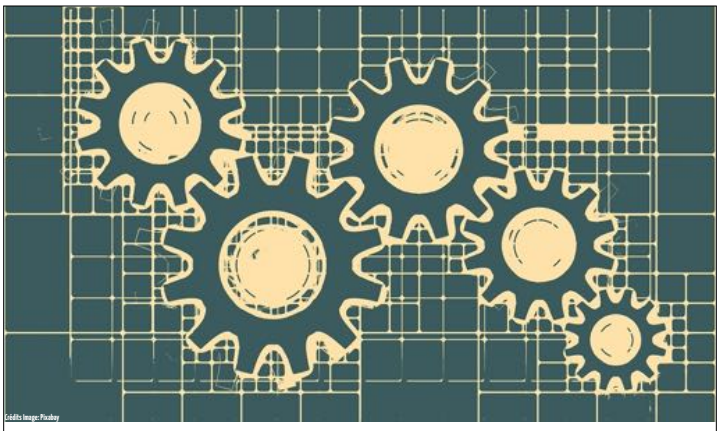
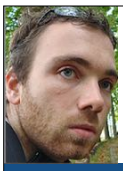


Crédits Image: Pixabay

Composition Logicielle et Passage à l'échelle

Sébastien Mosser
UQAM, 21.06.2018

Université Nice Sophia Antipolis
UNIVERSITÉ CÔTE D'AZUR

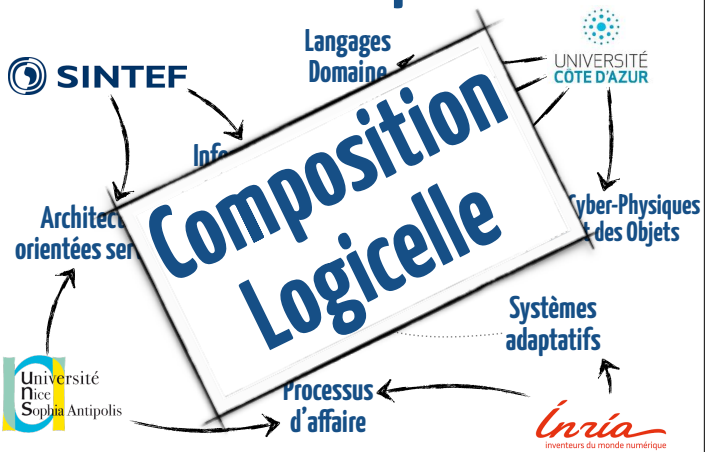


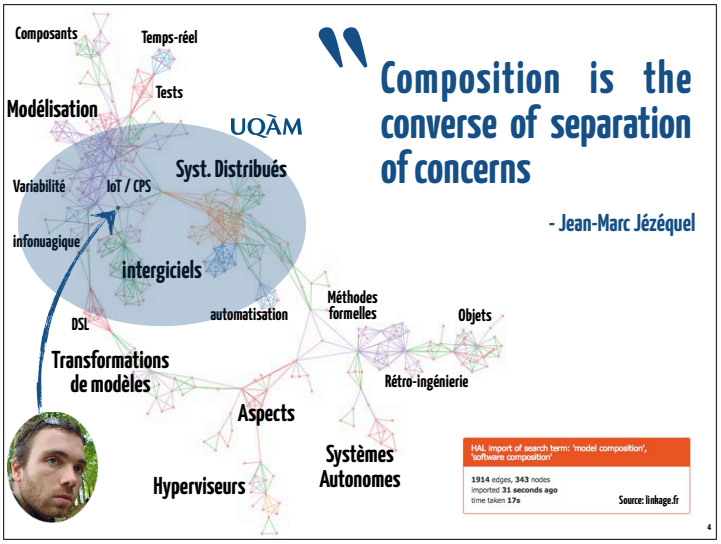
Sébastien Mosser

Université Côte d'Azur, Laboratoire I3S, Équipe SPARKS

- 12-... : Maître de Conférences, UCA
- 11-12 : Chargé de recherche (permanent), SINTEF (NO)
- 10-11 : Post-doc, Inria Lille Nord-Europe
- 07-10 : Doctorant (avec charge d'enseignement), UNS
- 04-07 : Élève-Ingénieur, Polytech Nice-Sophia

Carrière Académique





> "Composition is the converse of separation of concerns"
>
> - Jean-Marc Jézéquel

Étudiants de Doctorat (co-)encadrés

Qualité de Service & SOA (2010-2014)

Variabilité Logicielle - Matérielle (2016-...)

IHM & Capteurs IoT (2013-2017)

Méta-composition (2016-...)

Partage de réseaux de capteurs (2014-2017)

Systèmes Cyber-Physiques (2017-...)

De l'Industrie à la Recherche

Défis scientifiques issus de collaborations industrielles

Prototypes sous licences libre

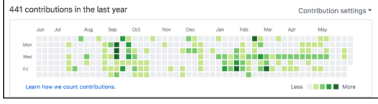
DATATHINGS

OSEAN
UNDERWATER TECHNOLOGY

TREEPTIK.
A LINKBYNET COMPANY

IBM

Meetup



AgilePlayGround.Org

1 Composition Logicielle ?

2 Composition & IoT

3 Composition & Micro-services

4 Composition & Réécriture

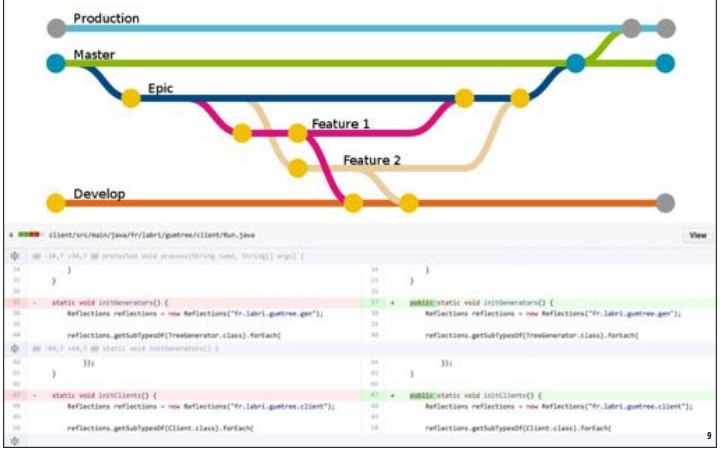
5 Vers un moteur de composition abstrait

Composition Logicielle

1

De quoi parle-t-on ?
Quels sont les défis ?

Intégration de code (git merge)



Fusion de paquetages (modèles)

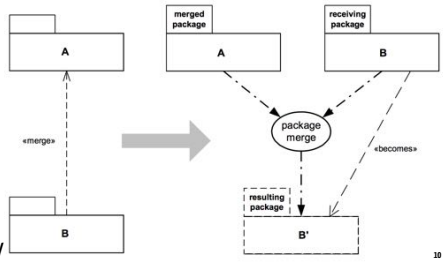
MOF2 and UML2 Superstructure uses Package Merge to define a number of different modeling languages and compliance levels

- Package merge was introduced to support metamodel reuse
- UML2 was partitioned to support Basic, Constructs, Kernel, and Levels L1, L2, and L3
- Package merge is used to construct the metamodel language at each level by reusing and merging packages

- Bran Selic, Jim Amsden, Kenn Hussey

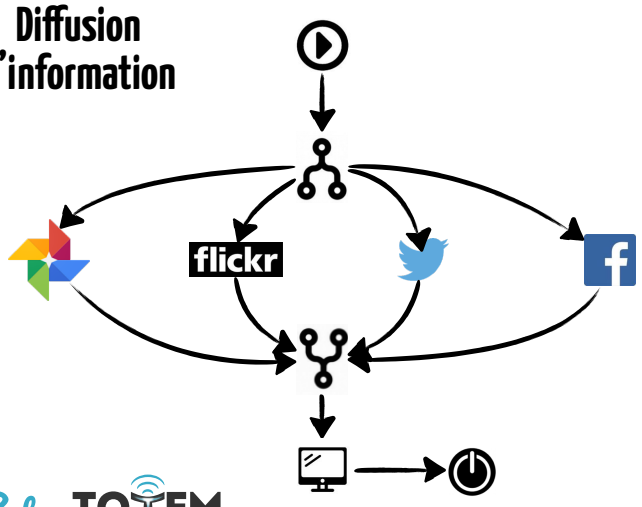
OBJECT MANAGEMENT GROUP

UNIFIED MODELING LANGUAGE



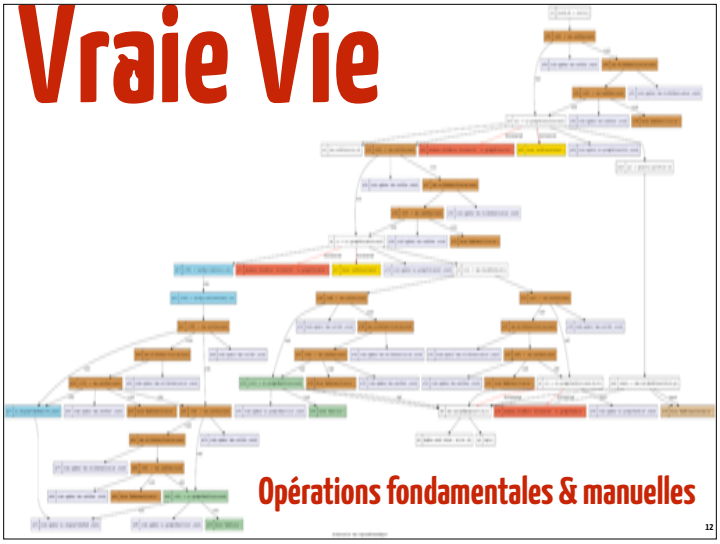
https://www.omg.org/spec/UML/2.5/About-UML/

Diffusion d'information



PulseTOTEM

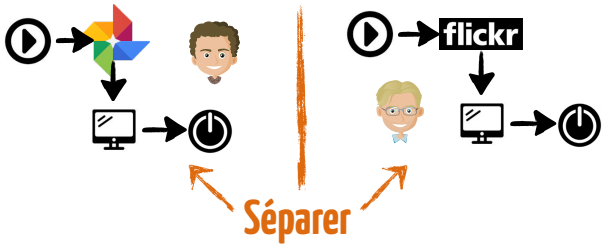
Vraie Vie

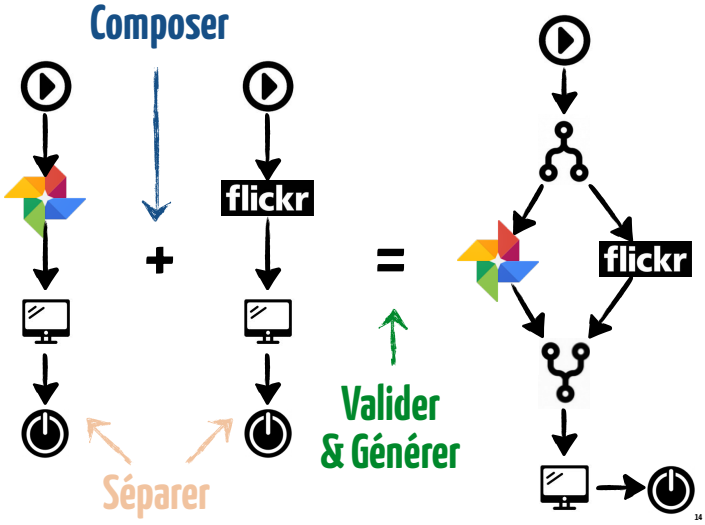


Opérations fondamentales & manuelles

"Build **complex things** by composing **small** and **simple** ones.

- E. Dijkstra [1972]



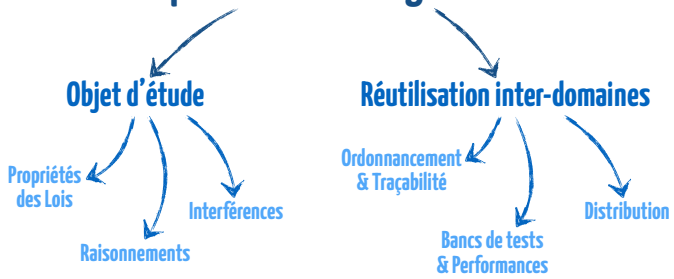


Collecte des besoins

M4S (2016); **FiaSCO** (2018)

Défis Identifiés

Composition & Passage à l'échelle



⇒ Rendre "composable" facilement un domaine

OSEAN
UNDERWATER TECHNOLOGY

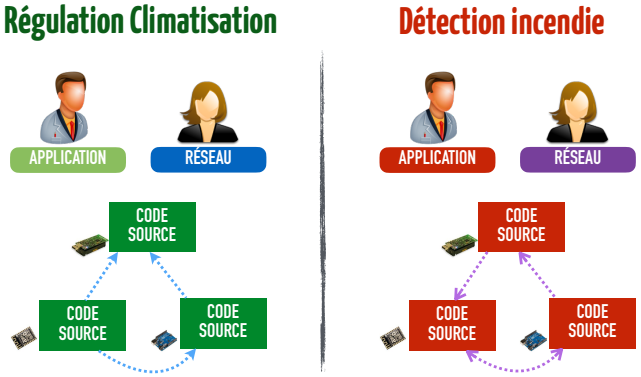
DATATHINGS

Systèmes Cyber-Physiques

2

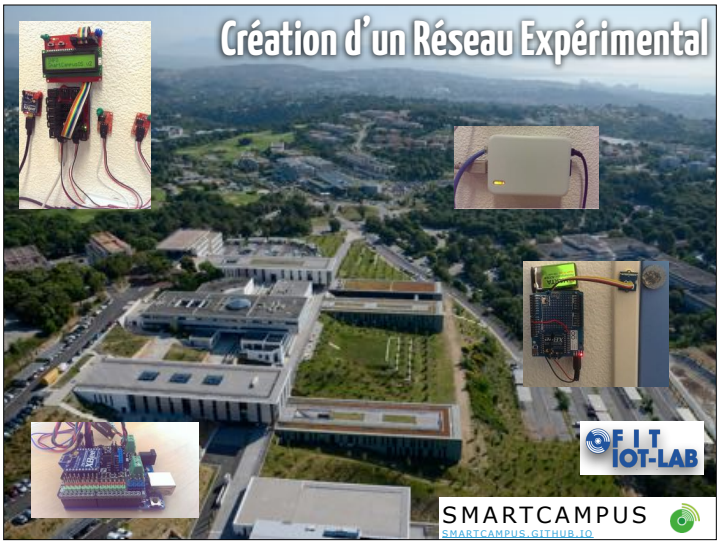
(Dé)Composition d'applications dans les réseaux de capteurs

Réutilisation & Réseaux de capteurs

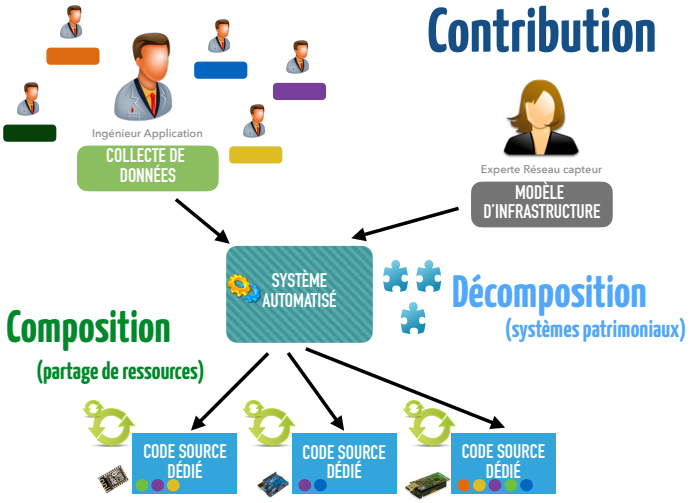


- Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M., & Polakos, P. (2015). Wireless Sensor Network Virtualization: A Survey.

Création d'un Réseau Expérimental



Contribution



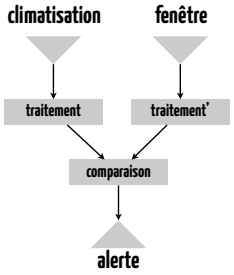
20

Collecte par flots de données

Structure de graphe

Sémantique formelle d'exécution

Vérification de propriétés de terminaison

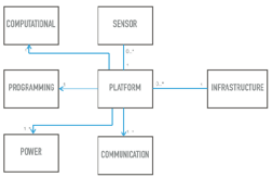


21

Modélisation de l'infrastructure

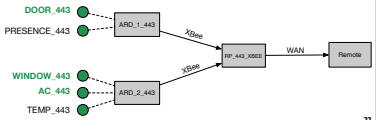
Variabilité des capacités du matériel

(UML + Modèle variabilité)



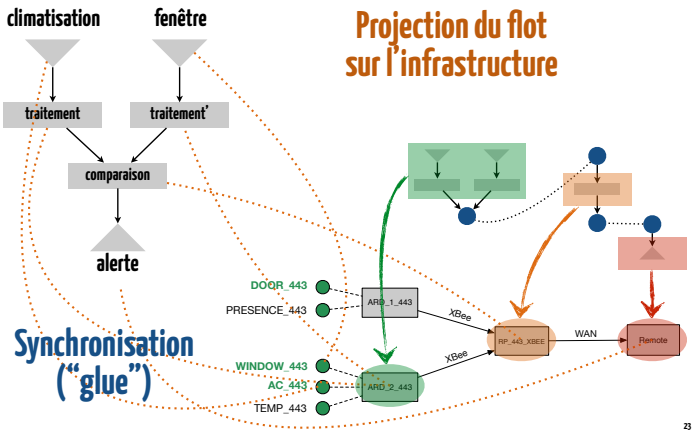
Topologie réseau

(Graphe orienté acyclique)



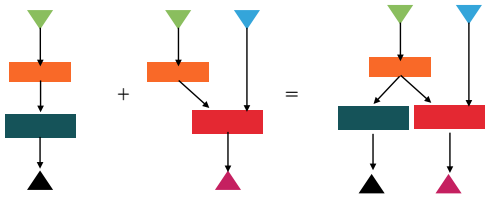
22

Réutilisation d'un réseau patrimonial



23

Réutilisation & Partage de ressources



Fusion de graphes par identification d'éléments équivalents

- Définition de **classes d'équivalences** entre activités
- **Ré-écriture du graphe** basée sur des actions élémentaires
- **Validation de propriétés** sur les ré-écritures :
  - Fonctionnelle (priorité, ...) & Non-fonctionnelle (absence d'accès concurrents, ...)

24

Validation fonctionnelle

(50 bureaux)

Détection Incendie

Travailleur isolé

Régulation Climatisation

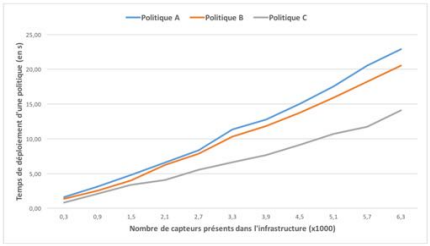
DEPLOYMENT OF THE RUNNING EXAMPLE (COMPREHENSIVE POLICY: 50 OFFICES)

	DEPOSIT source	# Generated files	# Generated LoC	# Concepts (before expansion)	# Concepts (after expansion)	Deployment time (in s)
Template	19	N/A	N/A	N/A	N/A	N/A
Single office	19	3	267	5	8	2.5
Comprehensive policy (without composition)	455	105	11685	250	400	50

SMARTCAMPUS

SMARTCAMPUS.GITHUB.IO

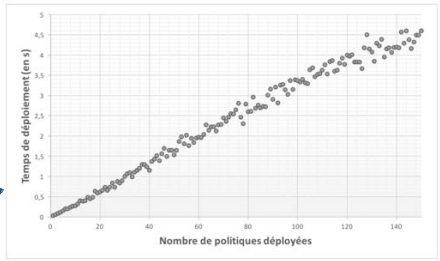
25



Performance & Complexité

70K capteurs (Moscou)

150 flots



26

Bilan

(Dé)Composition pour réseaux patrimoniaux

Recherche

Évaluation de performances

Enseignement

Maîtrise (Langages Dédiés)

Projet de Licence

Vérification de propriétés
Fonctionnelles & Non-Fonctionnelles

Industrie

DataThings (LU)
OSEAN (FR)

Publications

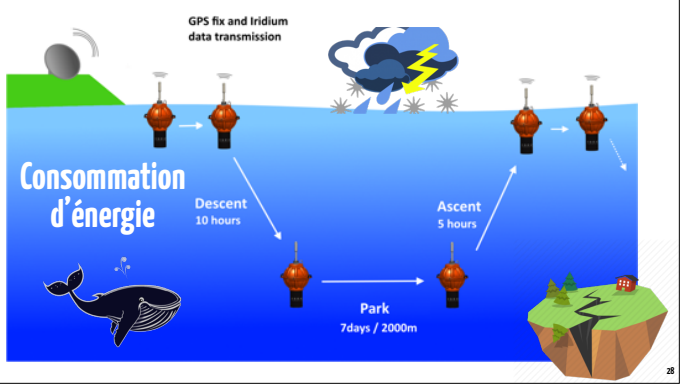
- APSEC'16
- SAC'16
- ICSR'15
- UMC'14
- (FGS en cours)

27

OSEAN UNDERWATER TECHNOLOGY — GeoAzur — erc

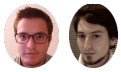
(Chercheur Principal : Guust Nolet)

GLOBALSEIS / MERMAID



28

IBM TREEPTIK. A LINKBYNET COMPANY

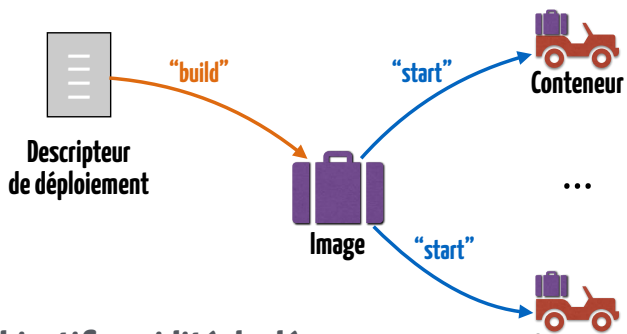


Déploiement de micro-services

3

Détection statique d'interactions de déploiement

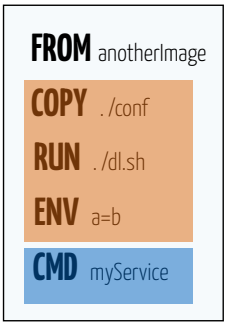
Déploiement de (micro-)Services



Objectif: rapidité de démarrage

30

Exemple de descripteur



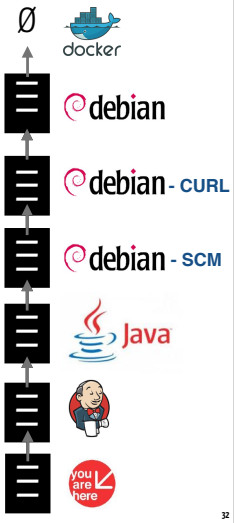
Réutilisation d'une image

Config. de l'environnement

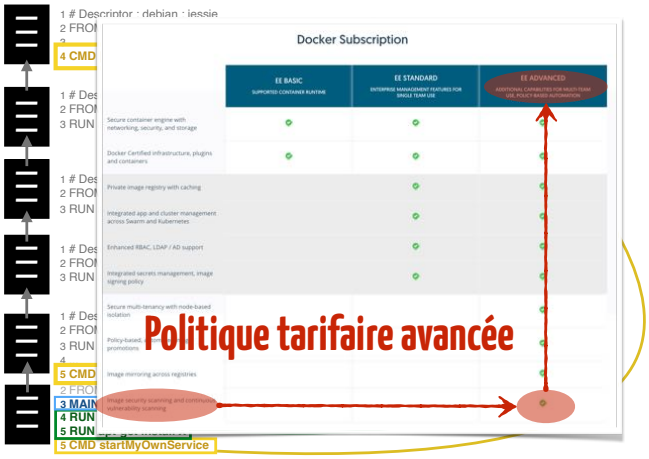
Démarrage du service

Composition Logicielle

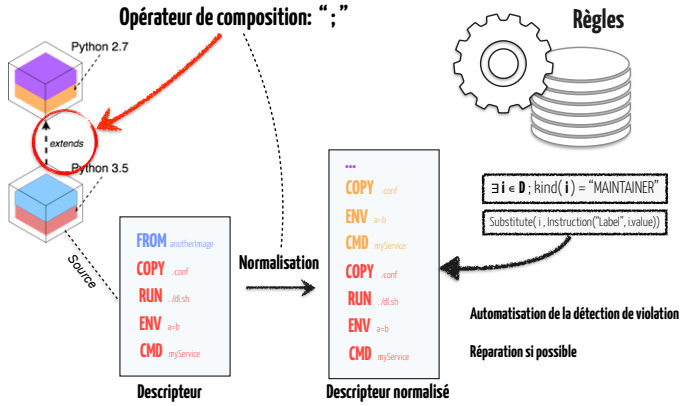
- Réutilisation d'images existante
  - "Boite noire"
  - Optimisation & Heuristiques
- Catalogue d'image à disposition
  - Images "vérifiées©"
  - Images "sauvages"



Et alors ? Où sont les problèmes ?



Contribution à l'écosystème Docker



Originalité vis à vis de l'état de l'art : **Exploitation de la relation de composition**

Formalisation des Descripteurs

$$d \in D = [i_1, \ldots, i_n] \in I^n_<$$
$$parent : D \to D$$
$$d \mapsto d' : d \text{ loads } d'$$

Séquence d'instructions ⊕ relation de parenté

$$;: D \times D \to D$$
$$(d_1, d_2) \mapsto d_{12} : \text{Let } d_1 = [i_1, \ldots, i_n],$$
$$d_2 = [i'_1, \ldots, i'_m],$$
$$d_{12} = d_1; d_2 = [i_1, i_n, i'_1, i'_m]$$
$$\wedge\, parent(d_2) = d_1$$
$$\wedge\, parent(d_{12}) = parent(d_1)$$

Composition par concaténation de séquences

Règles de détection de conflits

Modèle ouvert : les règles évoluent dans le temps

Abstraction via des formules de logique de premier ordre

$$\varphi_i : D \to \mathbb{B} \in \Phi$$
$$violation? : D \times \Phi^n \to \mathbb{B}$$
$$(d, \{\varphi_1, \ldots, \varphi_n\}) \mapsto \bigvee_{i=1}^{n} \varphi_i(\overline{d})$$

Réification de 19 bonnes pratiques Docker (2018)

$$\exists\, (i, i') \in \mathbb{D} ;\ kind(i) = kind(i') = \text{"CMD"} \wedge i < i'$$

Réparation des descripteurs

$$do : D \times \Sigma \to D$$
$$(d, \sigma) \mapsto d' : \text{Let } d = [\ldots, i_n, i, i_m, \ldots],\ \sigma = (i \to i'),\ d' = [\ldots, i_n, i', i_m, \ldots]$$

} Substitution de termes logiques

$$do^+ : D \times \Delta \to D$$
$$(d, \delta) \mapsto \begin{cases} \delta = \emptyset & \Rightarrow d \\ \delta = \{\sigma\} \cup \delta' & \Rightarrow do^+(do(d, \sigma), \delta') \end{cases}$$

} Application aux descripteurs

$$\text{conflict?} : \Delta \to \mathbb{B}$$
$$\delta \mapsto \text{Let } (i, a, b) \in \mathbb{I}^3, i \neq \emptyset, \exists (i \to a) \in \delta, (i \to b) \in \delta, a \neq b \neq \emptyset$$

$$\rho_i : D \to \Delta \in R$$
$$rw : D \times R^n \to D \,|\, Error$$
$$(d, rules) \mapsto d' : \text{Let } \delta = \bigcup_{\rho \in rules} \rho(\overline{d}),\ d' = \begin{cases} \text{conflict?}(\delta) & \Rightarrow Error \\ \neg\text{conflict?}(\delta) & \Rightarrow do^+(\overline{d}, \delta) \end{cases}$$

} **Ré-écriture automatique**


```
RUN npm install soap
RUN npm install json-schema-mapper
RUN npm install xml2js
```

problèmes d'installation

$$\rho_{pack} : D \to \Delta \in R$$
$$d \mapsto \delta : \text{Let } Is = \{i : i \in d, install?(i)\},\ \delta = \begin{cases} Is = \emptyset & \Rightarrow \emptyset \\ Is = \{i\} \cup Is' & \Rightarrow Rewritten \end{cases}$$
$$Removed = \{(i' \to \emptyset) : i' \in Is'\}$$
$$packages = \bigcup_{i \in Is} args(i) \backslash \{\texttt{npm}, \texttt{install}\}$$
$$m = run(\{\texttt{npm}, \texttt{install}\} \cup packages)$$
$$Rewritten = \{(i \to m)\} \cup Removed$$
$$install? : I_d \to \mathbb{B}$$
$$i \mapsto kind(i) = \text{RUN} \wedge \texttt{npm} \in args(i) \wedge \texttt{install} \in args(i)$$

```
RUN npm install json-schema-mapper soap xml2js
```


Bonnes Pratiques Officielles

Validation à large échelle

1. FROM command first
the FROM command must be the first to appear in a dockerfile

2. RUN Exec form
RUN commands have two syntaxes, one with brackets and one without. Interpretation of arguments differ from the two syntaxes. The one with brackets must be used.

3. Multiple CMD
CMD commands allows one to start a service when booting up a container. Docker allows only **a single service** to be specified, therefore multiple CMD are useless since only the last one will be run.

4. Provides default to CMD
One has to provide default parametervia CMD to start a service. If an EntryPoint command is specified, CMD and EntryPoint commands should be specified in JSON format.

5. Variables in exec form of CMD
Variables used in CMD commands in its exec form are not interpreted. CMD ["echo", "$HOME"] won't output the $HOMEvariable value.

6. Merge LABEL commands
When possible, merge labels commands.

7. Avoid apt-get upgrade
You should avoid RUN apt-get upgrade or dist-upgrade, as many of the "essential" packages from the base images won't upgrade inside an unprivileged container

8. Combine install with update
Always combine RUN apt-get update with apt-get install in the same RUN statement. Ommiting this can lead to unexpected behaviour since apt-get update can be not run.

9. Packages, version pinning
Always fully specify the version of the package to install.

10. FROM, version pinning
Always fully specify the version of the parent dockerfile to use (**i.e. latest** tag is therefore not permitted).

11. CMD exec form
CMD commands have two syntaxes, one with brackets and one without. Interpretation of arguments differ from the two syntaxes. The one with brackets must be used if parameters are specified.

12. Prefer COPY
Although ADD and COPY are functionally similar, generally speaking, COPY is preferred.

13. ADD <http> discouraged
Because image size matters, using ADD to fetch packages from remote URLs is strongly discouraged; you should use curl or wget.

14. User root discouraged
You should avoid installing or using sudo since it has unpredictable. TTY and signal-forwarding behavior that can cause more problems than it solves. If you absolutely need functionality similar to sudo (e.g., initializing the daemon as root but running it as non-root), you may be able to use gosu.

15. Less USER commands as possible
To reduce layers and complexity, avoid switching USER back and forth frequently.

16. WORKDIR must have absolute path
For clarity and reliability, you should always use absolute paths for your WORKDIR.

17. cd in RUN should be avoided
Don't use cd in RUN commands, use WORKDIR instead.

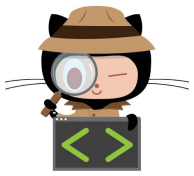
18. Sort installation alphanumerically
Installation of multiple softwares must be written in alphanumerical order.

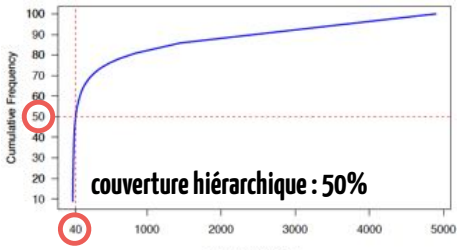
19. Add --no-install-recommend
Add --no-install-recommend when installing with apt-get, this will avoid installation not explicitly specified.
Guidelines can be found here. Guidelines 4 and 11 are not implemented since they are too domain-specific

Fouille des dépôts GitHub

22,945 descripteurs non triviaux

178K instructions



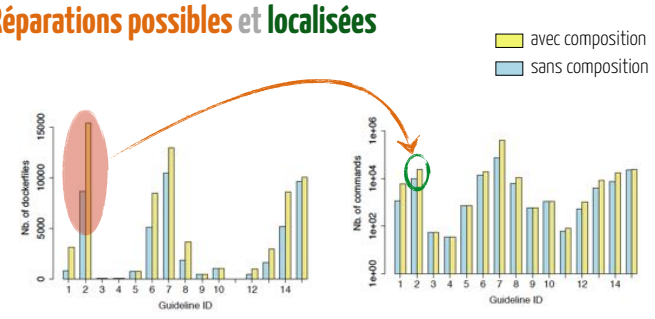


Résultats Expérimentaux

11K descripteurs analysés en 6s

Nombreuses interactions dans les hiérarchies

Réparations possibles et **localisées**



Bilan

Recherche

Formalisation d'un opérateur de composition

Détection d'interférences

Réparation automatique

Étude empirique (GitHub)

Enseignement

Maîtrise ("DevOps")

Spécialité (microservices)

Industrie

Treeptik, IBM
Docker Meetup
Amadeus

Publications

- SAC'18
- GlobalTech Amadeus
- EMCFA'14
- MODELS'12

UQÀM

Réécriture automatisée de code

4

Réécriture automatique "fiable" à large échelle

Contexte : Réécriture du noyau Linux

766,508 "commits"

19.06.18

```
…
x = kmalloc(sizeof(*a), 0);
memset(a, 0, sizeof(*a))
…
```

↓

```
…
x = kzalloc(sizeof(*a), 0);
…
```

Coccinelle (Lawall, Muller et al)

Principe de "patch" sémantique (Coccinelle)

```
@@
type T;
expression x, E, E1,E2;
@@
- x = kmalloc(E1,E2);
+ x = kzalloc(E1,E2);
... when != \( x[...]=E; \| x=E; \)
- memset((T) x, 0, E1);
```

$(a)\ \texttt{kmalloc}\wedge\texttt{memset}(0) \mapsto \texttt{kzalloc}\ \ (R_k)$

Réécriture : Programme → Programme

Composition : Réutilisation des réécritures

```
@@
type T;
expression x, E, E1,E2;
@@
- x = kmalloc(E1,E2);
+ x = kzalloc(E1,E2);
... when != \( x[...]=E; \| x=E; \)
- memset((T) x, 0, E1);
```

```
@@
type T;
T *x;
expression E;
@@
- memset(x, E, sizeof(x))
+ memset(x, E, sizeof(*x))
```

Règles de Réécriture

```
struct Point {
  double x;
  double y;
};
typedef struct Point Point;

int main()
{
  Point *a;
  // ...
  a = kmalloc(sizeof(*a), 0);
  // not using a
  memset(a, 0, sizeof(a));
  // ...
  return 0;
}
```

Code source cible

Comp…tures

Semantic patches and patches related to collateral evolutions

Patches that have been integrated into the official Linux tree (Linus Torvalds' repository):

- Use UPIO_MEM. 1 (semantic patch)
- Use resource_size. 1 2 3 4 5 6 (semantic patch)
- Use dev_get_drvdata 1 2 (semantic patch)
- use usb_get_intfdata, usb_set_intfdata. 1 2 (semantic patch)
- use usb_endpoint functions. 1 2 3 4 5 6 7 8 8 9 10 11 12 (semantic patch) 13 (semantic patch)
- use DEFINE_SPINLOCK. 1 2 3 4 5 5 (semantic patch)
- use ARRAY_SIZE. 1 2 3 4 5 (semantic patch)
- use DIV_ROUND_CLOSEST. 1 2 3 4 (semantic patch) 4 (semantic patch)
- use DIV_ROUND_UP. 1 2 3 4 5 6 7 8 (semantic patch) 8 9 10 11 12 (semantic patch)
- use BUG_ON. 1 2 3 4 5 6 7 8 (semantic patch)
- use FIELD_SIZEOF. 1 (semantic patch)
- use time_before, time_before_eq, etc. 1 2 3 4 5 6 7 8 9 10 11 12 13 (semantic patch)
- potential-error-irqfunc.patch (semantic patch)
- kzalloc-treewide.patch (semantic patch)

Bug fixing patches

- Use kzalloc for allocating only one thing 1 2 (semantic match)
- Fix size given to memset 1 2 3 4 5 6 (semantic match)
- Size of a pointer 1 2 3 4 5 6 7 8 (semantic match)
- Use PTR_ERR to get error code 1 (semantic match)
- Use ERR_PTR/IS_ERR to return a flag as a pointer 1 (semantic match)
- Remove exceptional

59! > nombre d'atomes dans l'univers

R2

P'

Composition : Réutilisation des réécritures

R1

R2

?

P

R2(R1(P)) = P''

Pas d'erreurs au sens de Coccinelle

1ère étape : Un modèle pour raisonner

$$\text{Let } \rho = (\varphi, \chi) \in (\Phi \times X) = P, \quad (\varphi : AST \to AST) \in \Phi$$
$$\chi : AST \times AST \to \mathbb{B} \in X, \qquad \forall p \in AST,\ \chi(p, \varphi(p))$$

Indépendance Technologique

$$apply : AST \times P^n_< \to AST$$
$$p, [\rho_1, \ldots, \rho_n] \mapsto \text{Let } p_{2..n} = (\bullet^n_{i=2}\varphi_i)(p),\ p' = \varphi_1(p_{2..n}),\quad \chi_1(p_{2..n}, p')$$

Isofonctionnalité avec l'outillage existant

2nde étape : Détecter les problèmes

$$seq : AST \times P^n_< \to AST$$
$$p, [\rho_1, \ldots, \rho_n] \mapsto p_{seq} = (\bullet^n_{i=1}\varphi_i)(p),$$

$\wedge^n_{i=1} \chi_i(p, p_{seq})$

R2(P'') = P''
R1(P'') ≠ P''

R2(P') = P'
R1(P') = P'

Trouver l'ordre est un problème ... Et si on pouvait s'en passer ?

$$iso : AST \times P^n \to AST$$
$$p, \{\rho_1, \ldots, \rho_n\} \mapsto p_{iso} = p \oplus (\dot{}^n_{i=1}(\varphi_i(p) \ominus p)),\quad \wedge^n_{i=1} \chi_i(p, p_{iso})$$

$$\oplus : AST \times A^*_< \to AST$$
$$(p, S) \mapsto \begin{cases} S = \varnothing & \Rightarrow p \\ S = \alpha \,|\, S' & \Rightarrow exec(\alpha, p) \oplus S' \end{cases}$$
$$\ominus : AST \times AST \to A^*_<$$
$$(p', p) \mapsto \Delta, \text{ where } p' = p \oplus \Delta$$

Hypothèse : Opérateur de Diff

Validation

RunnerUp
(https://github.com/jonasoreland/runnerup)

4 règles de **réparation énergétiques**

24 séquences différentes

Composition isolée
Composition séquentielle

Validation sur le noyau Linux en cours (extension journal)

Bilan

Modèle fonctionnel & logique

Recherche

Ré-agencement des abstractions

Propriétés de composition

Validation au niveau du code

Enseignement

Licence (Tests par mutations)

Maîtrise ("DevOps")

Industrie

Android
Linux

Publications

- ICSR'18
- JSS (en cours)
- SoSym (en cours)

Moteur de Composition Abstrait & Passage à l'échelle

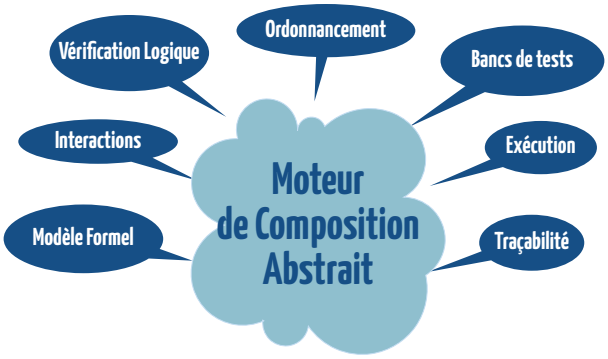
5

Proposition de collaboration

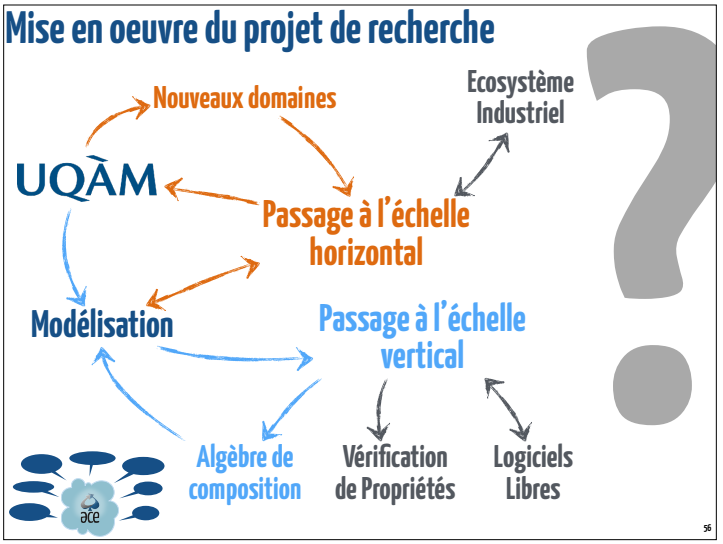
UQÀM

Projet de Recherche

http://ace-design.github.io/



Mise en oeuvre du projet de recherche



Recherche Subventionnée (en volume)

	En Cours	Terminé	Total
Institutionel	105 K€	460 K€	565 K€
Industriel	167 K€	58 K€	225 K€
Collaboratif	61 K€	20 K€	81 K€
Total	333 K€	538 K€	

(ne sont pas comptés les projets collaboratifs FP7,H2020)

Activités d'Animation

- **Recherche** (national)
  - Co-responsable de GL/ΛCE pour le CNRS / GdR GPL (Systèmes Cyber-Physique)
  - Responsable de l'action PING (Enseignement du GL en Maîtrise et Doctorat)
  - Action de formation des ingénieurs du CNRS (DEVLOG)
- **Enseignement**
  - Co-fondateur de la Nuit de l'Informatique (2007 - ...)
  - Co-fondateur de l'Agile Playground Sophia Antipolis (2013-2017)
  - Participation à des rencontres d'intérêt
    - Docker, Software Craftsmanship, Agilité
  - Hackathon (non industriels)
    - Startup Week-End, Global Day of Code Retreat, Google HashCode