EmbedDSU : un Framework de HotSwUp pour Cartes à puce Java

Agn s C. Noubissi, Juli n Iguchi-Cartigny t J an-Louis Lan t

Univ rsité d Limog s,
83 ru d'Isl ,
87000, Limog s - Franc
agn s.noubissi@xlim.fr, juli n.cartigny@xlim.fr, j an-louis.lan t@xlim.fr

Résumé
Dans l but d corrig r l s vulnérabilités, d r par r d s bugs ou d'amélior r d s fonctionnalités, l s applications ou l s systèm s sont app lés à évolu r, t donc à êtr mis à jour. L proc ssus d mis à jour traditionn l consist à stopp r l'application ou l systèm compl t, puis d'appliqu r la mis à jour t nfin d r démarr r l s élém nt ét ints. C p ndant c tt approch n' st pas adapté pour c rtains systèm s mbarqués (bancair s, télécommunication, tc.) où il st néc ssair d n pas stopp r l s s rvic s off rts mêm durant la mis à jour.
Un solution approprié st la mis à jour dynamiqu (Hot Softwar Updat : HotSwUp). C ll -ci consist à appliqu r la mis à jour n lign c' st-à-dir sans stopp r l' xécution du composant. Dans la littératur , d nombr ux travaux ont été ff ctués dans c domain pour l s systèm s natifs ou l s systèm s s' xécutant au-d ssus d'un machin virtu ll . ais c s travaux ont été réalisés pour d s configurations comm d s post s d travail ou d s s rv urs. La mis n plac d'un mécanism d HotSwUp pour l s p tits obj ts mbarqués notamm nt l s cart s à puc , r st ncor un défi au r gard d s contraint s d r ssourc s t d sécurité.
Dans c t articl , nous prés ntons Emb dDSU, un systèm d mis à jour dynamiqu d s composants systèm s Java pour cart à puc , basé sur un machin virtu ll Java (Java Card). L'idé étant d limit r l proc ssus d mis à jour uniqu m nt aux parti s aff cté s par la modification afin d réduir la compl xité d l'opération dans la cart . Emb dDSU st basé sur un ns mbl d mécanism à l' xtéri ur d la cart (off-card) t à l'intéri ur d la cart (on-card). En off-card, il s'agit d dét rmin r l s différ nc s syntaxiqu s ntr d ux class s, d' xprim r c ll s-ci dans un langag spécifiqu conçu à c t ff t, t d transfér r à la cart , l fichi r résultant (DIFF). En on-card, il s'agit d'int rprét r l fichi r d DIFF t d m ttr à jour la définition d la class , l s instanc s d'obj ts t l s autr s structur s d donné s aff cté s.

Mots-clés : Java Card, mis à jour dynamiqu , machin virtu ll Java, cart à puc

1. INTRODUCTION

L s applications sont app lé s à évolu r dans l but d fix r d s défauts ou d s bugs, d'amélior r t/ou supprim r d s fonctionnalités. Un approch simpl d mis à jour consist à stopp r l'application ou l systèm à m ttr à jour, d'appliqu r la mis à jour t d r démarr r la nouv ll v rsion. C p ndant, c tt approch n' st pas approprié pour l s applications néc ssitant d s' xécut r continu ll m nt mêm durant la mis à jour. Un solution adapté st la mis à jour dynamiqu (HotSwUp) qui consist à m ttr à jour un application sans stopp r c tt d rnièr , ni stopp r l systèm sur l qu l ll s' xécut . Dans c papi r, nous prés ntons Emb dDSU, un approch d mis à jour d s applications t/ou composants systèm s sur un cart à puc d typ Java Card. L'approch st basé sur la modification d la machin virtu ll Java mbarqué dans l'obj ctif d'offrir l s fonctionnalités d mis à jour dynamiqu . Emb dDSU st divisé n d ux parti s : off-card t on-card. En off-card, dans l but d'appliqu r la mis à jour uniqu m nt sur l s parti s aff cté s par la modification, un générat ur d DIFF a été implém nté. L générat ur d DIFF dét rmin l s modifications syntaxiqu s ntr d ux v rsions d class s t xprim l s différ nc s trouvé s dans un langag dédié. n obti nt ainsi un fichi r DIFF. C fichi r st nsuit transféré dans la cart t utilisé pour la mis à jour.

Sous c rtain s hypothès s spécifiqu s - notamm nt qu Emb dDSU fonctionn av c l s composants pr s ntants un équival nc hiérarchiqu (arbr d'héritag) ntr l'anci nn t la nouv ll v rsion -, l'approch p rm t d s mis à jour génériqu s d s applications t composants systèm s Java t possèd l s propriétés suivant s :

1. Pas d'int rv ntion humain : l proc ssus st total m nt automatiqu . En ff t, étant donné s d ux v rsions d'un class , l générat ur d DIFF st capabl d dét rmin r l s différ nc s syntaxiqu s ntr l s d ux v rsions au niv au d s int rfac s, champs, signatur s d méthod s, instructions, t d s métadonné s d s class s t ll s qu l s pools d constant , tc.
2. Support d plusi urs granularités int rn s d mis à jour : la mis à jour p ut êtr ff ctué au niv au d s champs d la class , d s signatur s d s méthod s, d s blocs d'instructions t d s class s r lativ s.
3. Atomicité d la mis à jour : dans l but d'assur r la cohér nc du systèm , la mis à jour st appliqué d façon atomiqu (un mécanism d roll-back st prévu).
4. Prés rvation d la sémantiqu : après la mis à jour, l s nouv ll s v rsions d s class s obt nu s n utilisant l fichi r d DIFF ont l mêm comport m nt qu si ll s avai nt été transféré s t qu la résolution t l'édition d s li ns ai nt été ff ctués dans la cart .

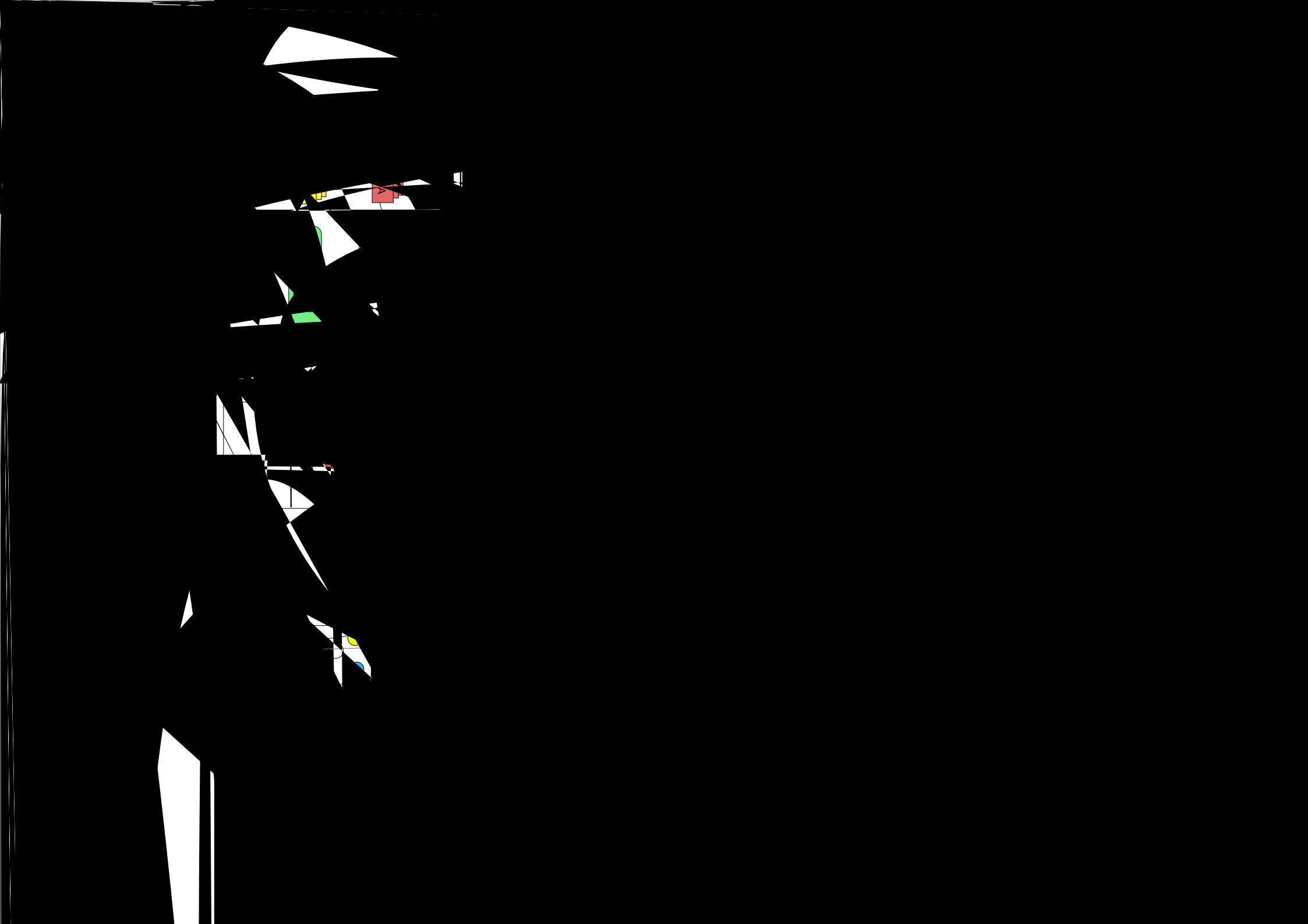
Pour valid r notr approch , nous avons implém nté t t sté Emb dDSU à l'aid d Simpl RTJ [19], un machin virtu ll Java pour l' mbarqué. Nous avons modifié Simpl RTJ afin d'offrir l s mécanism s d mis à jour dynamiqu . C t articl prés nt notr motivation, xpliqu Emb dDSU, décrit son implém ntation, prés nt l s résultats préliminair s obt nus t dév lopp l s travaux futurs.

2. MOTIVATION

Un cart à puc st un ordinat ur d la taill t form d'un cart d crédit, sur l qu l st mbarqué un ou plusi urs microcontrôl urs. Général m nt, l s microcontrôl urs pour cart s possèd nt un microproc ss ur possédant plusi urs typ s d mémoir s : RA (pour stock r l s donné s t l s pil s d thr ad durant l' xécution), R (dans l qu l l systèm d' xploitation t l s applications sont gravé s) t EEPR (pour l stockag d s donné s p rsistant s). En raison d s contraint s d taill d la puc , la taill d la mémoir st faibl . La plupart d s cart s à puc v ndu s sur l marché aujourd'hui ont au plus 5 Ko d RA , 256 Ko d R t 256 Ko d'EEPR . La communication ntr l l ct ur t la cart st réalisé par un liaison séri Half dupl x av c un protocol app lé APDU (Application Protocol Data Unit). La cart agit comm un s rv ur t att nd général m nt l s r quêt s nvoyé s par l l ct ur.

L s cart s à puc mod rn s intègr nt un machin virtu ll Java qui int rprèt l s applications préalabl m nt gravé s dans la R t aussi c ll s qui sont téléchargé s dans l'EEPR après émission d la cart (opération d Post-issuanc). La norm Java Card 3.0 [6, 7, 8] st un x mpl d spécification d la machin virtu ll pour cart à puc , disponibl n d ux éditions : édition classiqu ou édition conn cté . La machin virtu ll Java Card (JCV) int rprèt l s applications Java t gèr l'accès aux r ssourc s d la cart à puc , s rvant ainsi d systèm d' xploitation à la cart . La JCV st un machin virtu ll qui un fois instancié sur la cart , d vi nt un application qui n s'arrêt jamais. En ff t, sa duré d vi st similair à c ll d la cart , car l s obj ts p rsistants sont cons rvés mêm après l' xpiration d s s ssions d communication av c l l ct ur. Ainsi, la JCV fonctionn n p rman nc t la mis à jour d s composants doit êtr réalisé durant l' xécution.

La sécurité d la plat form du point d vu logici l r pos sur plusi urs mécanism s. Tout d'abord, la possibilité d télécharg r un application dans la cart st contrôlé par un protocol défini par Global Platform [18]. C protocol garantit qu l propriétair d l'application possèd l s droits néc ssair s pour ff ctu r un t ll action. Il définit aussi d s domain s d sécurité pour chaqu fourniss ur d'application. En ff t, Java Card st un plat form ouv rt , c' st-à-dir , qu' ll offr la possibilité d charg r t d' xécut r d nouv ll s applications après émission d la cart , t d s applications appart nant à plusi urs fourniss urs p uv nt co xist r sur la mêm cart . Grâc à la vérification du systèm d typ , l s byt cod s produits par l compilat ur sont sans dang r c' st-à-dir qu l'application chargé n p ut pas int rfér r av c d'autr s applications. En outr , la Java Card possèd un par -f u dont l but st d contrôl r l s autorisations ntr l s applications d s domain s d sécurité différ nts t l'isol m nt d c ll s-ci lors d l' xécution.



L génératur d DIFF compar syntaxiqu m nt d ux v rsions d class s t fournit un spécification pour la mis à jour. L fichi r d DIFF résultant p ut comport r plusi urs parti s notamm nt pour un class donné , l s int rfac s, l s champs t l s méthod s r lativ s aux modifications constaté s ntr l s d ux v rsions. C s spécifications sont utilisé s à l'intéri ur d la cart pour :

– transform r l cod d s class s, l s obj ts dans l tas, t l s fram s dans la pil d thr ad pour obt nir c ux corr spondants à la nouv ll v rsion,
– dét rmin r l s points du systèm dit 'sûrs', durant l qu l la mis à jour p ut êtr ff ctué n cons rvant la cohér nc du systèm .

Dét ct r un point du systèm dit 'sûrs', pass par la r striction d c rtain s méthod s à apparaitr dans la pil d thr ad. C s méthod s r str int s p uv nt êtr , pour un class donné : (1) s s méthod s modifié s, (2) t l s méthod s d s class s r lativ s impacté s par la mis à jour comm décrit plus loin dans la s ction 3.3.2.

3.2. Types de mise à jour supportés

Emb dDSU support l s modèl s d mis à jour l s plus fréqu nts, pour un class donné . Il s'agit d la modification :

– mbr s donné s : un mis à jour p ut port r sur la modification d s champs statiqu s ou d'instanc s d'un class . Il p ut s'agir plus précisém nt d la mis à jour d s modificat urs (*public, private, protected, etc*), d s typ s, d l'ordr , ou mêm just d s noms d s m mbr s donné s d'un class .
– Signatur d méthod s : il s'agit d la modification du typ t du nombr d'argum nts d s méthod s d la class . La modification d la signatur d'un méthod d'un class impact l s autr s class s n r lation av c c tt d rnièr . En ff t, c s class s p uv nt fair app l à c s méthod s dont la signatur a changé, t ll s doiv nt donc êtr aussi mis s à jour.
– Variabl local d s méthod s : bi n qu l s ul modificat ur autorisé pour l s variabl s local s st *final*, d'autr s typ s d modifications p uv nt surv nir notamm nt au niv au du typ d la variabl .
– Bloc d'instructions : la modification du corps d s méthod s st l modèl d mis à jour l plus couramm nt pris n charg par l s systèm s d mis à jour xistants.

3.3. Implémentation

3.3.1. A l'extérieur de la carte

La pr mièr étap consist à calcul r l s chang m nts ntr d ux v rsions (l'original t la nouv ll) du modul systèm à m ttr à jour dans l but d dét rmin r l s modifications ff ctué s. Emb dDSU utilis , pour produir un fichi r d DIFF, un génératur d DIFF [1, 2]. C fichi r conti nt l s modifications syntaxiqu s ntr l s d ux v rsions, xprimé s dans un langag dédié conçu à c t ff t. C fichi r st nsuit int rprété t formaté dans un format binair afin d réduir la taill t réduir aussi l coût d l'int rprétation dans la cart . Enfin, l fichi r binair st transféré à la cart pour réalis r la modification d s structur s d donné s t informations approprié s pour un mis à jour ff ctiv .

3.3.2. A l'intérieur de la carte

Emb dDSU st basé sur la modification d la machin virtu ll Simpl RTJ dans l but d'offrir d s mécanism s d mis à jour dynamiqu . C tt modification inclut l'ajout d plusi urs modul s t ls qu :

1. odul d'int rprétation du fichi r d DIFF qui p rm t d'int rprét r l fichi r d DIFF t d'initialis r l s structur s d donné s néc ssair s à la mis à jour. C ll s-ci s ront par la suit utilisé s par l s autr s modul s.
2. odul d'introsp ction qui fournit d s fonctions d r ch rch à trav rs l s structur s d donné s d la machin virtu ll t ll s qu la tabl d référ nc , la tabl d thr ads, la tabl d class s, la tabl d s obj ts statiqu s, l tas d la machin virtu ll t bi n sûr la pil d thr ad. L'obj ctif étant d récupér r l s informations néc ssair s à d'autr s modul s.
3. odul d dét ction d *Safe point* p rm ttant d dét ct r l'instant auqu l on p ut appliqu r la mis à jour n prés rvant la cohér nc du systèm .
4. odul d mis à jour, modul p rm ttant d modifi r l s instanc s d'obj ts dans l tas, modifi r l s blocs d'instructions dans l corps d s méthod s, l s métadonné s d s class s, l s référ nc s d'obj ts dans la tabl d référ nc , l s r gistr s aff ctés dans la pil d thr ad t d'autr s structur s d donné s aff cté s à la machin virtu ll .

rm t d'obt nir la v rsion précéd nt du cod , d s in-
ur s d donné s d la machin virtu ll n cas d non-
a cart dans l l ct ur par x mpl).
s fonctionnalités d mis à jour dynamiqu durant l' xé-

pour qu' ll puiss fonctionn r sous trois mod s d' xé-

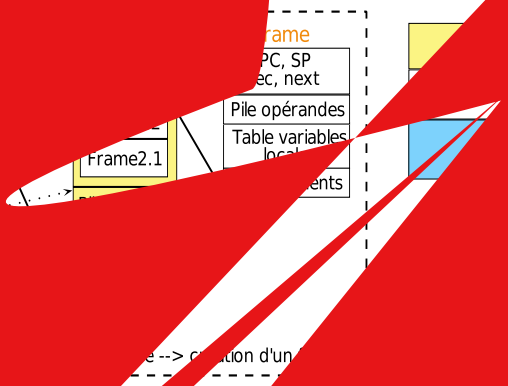
l avant la dét ction d'un nouv ll v rsion d l'applica-
r d DIFF,
d' xécution après réc ption du fichi r d DIFF t avant

n du *safe point* durant l qu l la mis à jour st ff ctué .

machin virtu ll . Dans c mod , la machin virtu ll
ir dû au proc ssus d HotSwUp. Après réc ption d'un
d cart , t après vérification t int rprétation du fichi r,
n rch d'un point 'sûr' du systèm .

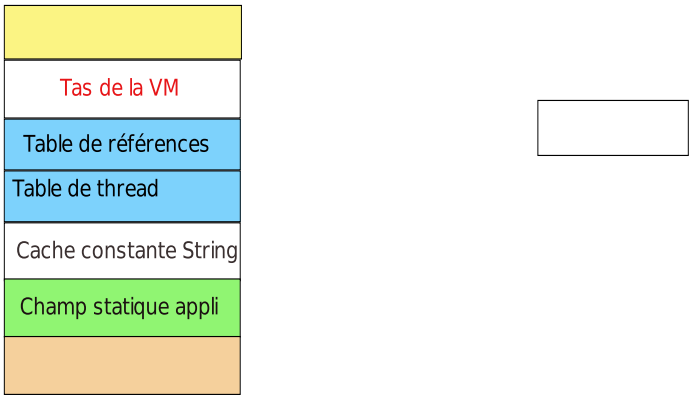
rosp ct r l'état d' xécution du composant à m ttr
mposé d :
f cté s par la mis à jour (l s byt cod s d s

fram s, r lativ s aux méthod s à m t
socié st créé dans la pil d thr ad
s, l r gistr d compt ur d'instr
c s activ s d la class à m ttr
s corr spondant à c ll s qui
s par un champ d'un aut




Comp
Métadonnées
class A
class B
Classe
Pool de constantes

4. is à jour d s fram s



Initial m nt	Champs r -ordonnés	Ajout champs	Suppr ssion champs

Class C {
int var1 ;
int var2 ;
short var3 ;
bj ct var4 ;
}
	class C' {
	int var2 ;
	short var3 ;
	int var1 ;
	bj ct var4 ;
	}
		class C' {
		short var5 ;
		int var1 ;
		int var6 ;
		int var2 ;
		short var3 ;
		bj ct var4 ; }
			class C' {
			int var1 ;
			short var3 ;
			}

TABLE 1 – L s v rsions d'instanc s utilisé s

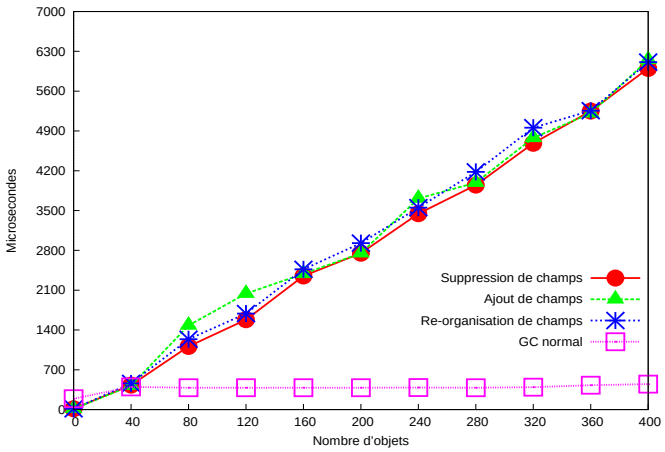


FIGURE 6 – Résultats d mis à jour obt nus n modifiant l s champs d'un class , comparé au GC

L GC normal st xécuté sans obj ts *morts* dans l tas d la machin virtu ll , donc sans surcoût supplém ntair dû par la libération d la mémoir . Dans c cas, l GC st optimisé tandis qu la mis à jour d s instanc s ont d s surcoûts supplém ntair s dûs aux fonctions t ls qu la création t l'initialisation d s nouv ll s instanc s d la class av c l s val urs fourni s n off-card t prés nt s dans l fichi r d DIFF. C s fonctions r nd nt l proc ssus plus coût ux par rapport à c GC normal.

D'autr part, on constat qu l'ajout d s champs st l'opération la plus coût us comparé aux autr s typ s d modifications, c ci dû au coût d la r ch rch d'un spac mémoir approprié d taill corr - spondant à c ll d'un instanc d la nouv ll v rsion d la class , à c la s'ajout l coût d la copi d s anci ns champs t l'initialisation d s nouv aux champs par l s val urs initial s fourni s prov nant du fichi r d DIFF.

La réorganisation d s champs st égal m nt coût ux mais moins coût ux par rapport à l'opération d'ajout d champs, c ci dû au fait qu la l ctur d s val urs initial s fourni s par l fichi r d DIFF n' st pas réalisé .

La suppr ssion d s champs r st l'opération la moins coût us , l s champs non supprimés sont copiés dans l nouv l spac mémoir rés rvé à la nouv ll instanc , t c tt r copi st réalisé n om ttant l s champs supprimés, n plus l'initialisation d s champs av c l s val urs fourni s par l fichi r d DIFF n' st ff ctué qu dans c rtains cas.

5. TRAVAUX RELATIFS

D nombr ux systèm s d mis à jour dynamiqu ont été réalisés. Notamm nt, dans l domain Java, on p ut cit r ntr autr s : JV LVE [9], JDRU S [10], DV [11] t l s travaux d' rso t Al. [10].

rso t An. [12] propos nt d'implém nt r la DSU sans modifi r la machin virtu ll n utilisant un

proxy pour chaqu class à m ttr à jour. Pour chaqu class C, un class proxy CP st proposé . Tous l s app ls v rs l s méthod s d C pass nt un app l dans CP qui lui s charg d'app l r la méthod d C. Ainsi, la mis à jour d'un class pass par la mis à jour d la class proxy t donc un app l d'un méthod d C, pass ra par CP qui app ll ra à son tour, la méthod corr spondant dans C'. Et d nouv aux obj ts d C' sont créés t mis à jour grâc aux anci nn s instanc s d C. C tt approch néc ssit à la class d' xport r l s mêm s int rfac s publiqu s t r str int la mis à jour. En ff t, on n p ut ajout r d'app ls à d s méthod s publiqu s qui n sont pas prés nt s dans l s int rfac s importé s dans l'anci nn v rsion.

DV [11] t JDRU S [10] implém nt nt la mis à jour dynamiqu n modifiant la machin virtu ll mais pas d façon similair à Emb dDSU. En ff t, l s aut urs d DV propos nt d'ét ndr l charg ur d class Java pour p rm ttr l r mplac m nt d'un définition d la class t la mis à jour d s obj ts instanciés. Pour c la, ils définiss nt d ux méthod s principal s : r loadClass () t r plac Class () p rm ttant r sp ctiv m nt d charg r la nouv ll v rsion d la class t d m ttr à jour l s instanc s d la class n cours. JDRU S, quant à lui, utilis l s tabl s d'indir ctions pour r mplac r l s anci nn s v rsions d s class s par l s nouv ll s n modifiant l s adr ss s d location d s class s. L principal inconvéni nt d c s t chniqu s st qu'on obs rv un surcoût mêm durant l mod standard d la machin virtu ll . Dans Emb dDSU, n mod normal d la machin virtu ll , aucun coût supplém ntair n' st obs rvé.

JV LVES [9] st un systèm implém ntant la DSU grâc à Jik s RV , un machin virtu ll ét ndu intégrant d s s rvic s d mis s à jour d s applications Java. Il prés nt un int rfac similair à Emb dDSU, n c s ns où, il possèd un outil app lé UPT (*Update Preparation Tool*) p rm ttant d dét rmin r ntr l'anci nn v rsion t la nouv ll v rsion d'un application :
– l s fichi rs class s modifiés,
– l s fichi rs class s supprimés,
– t c ux ajoutés.
Et à partir d là, la mis à jour n' st ff ctué qu'av c l s fichi rs class s modifiés t ajoutés. En plus, ll associ un fichi r p rm ttant la transformation d'état pour la mis à jour d s instanc s. Emb dDSU st proch d JV LVE par contr au li u d travaill r au niv au fichi r, nous avons travaillé au niv au cod av c pour granularité, la class . Ainsi, au li u d compar r d ux applications pour dét ct r l s fichi rs modifiés, nous avons comparé d ux class s pour dét rmin r l s modifications int rn s plus précisém nt l s modifications syntaxiqu s. L'obj ctif étant d transfér r un quantité réduit d'information au vu d s contraint s d r ssourc s n t rm d stockag sur la cart . En plus, JV LVE ajout un compilat ur JIT (Just-In-Tim), c qui augm nt l s coûts d mis à jour.
Général m nt, c s implém ntations sont d stiné s à d dém t form s possédant l s caractéristiqu s min-

sion d l'évolution d s chang m nts d l urs applications ou composants applicatifs.

6.2. Méthodes proxies

Emb dDSU p rm t un mis à jour d'un class t l s class s aff cté s d façon atomiqu . S'il y a un nombr important d class s à m ttr à jour, c tt approch p ut r ndr indisponibl la cart durant un c rtain mom nt (l t mps d'att nt maximal pour un utilisat ur d cart étant nviron d 3 s cond s). Par conséqu nt, l'idé st d s p nch r sur l s class s aff cté s, d fair un mis à jour total d s class s aff cté s possédant d s méthod s supprimé s t ff ctu r un mis à jour parti ll d c ll s cont nant uniqu m nt d s méthod s modifié s, n particuli r c ll s dont la signatur a été modifié . Et pour c la, fournir d s méthod s proxi s p rm ttant d pass r d'un app l d méthod dont la signatur a changé v rs la nouv ll v rsion d la méthod sans modifi r l cod d l'app l au niv au d la class aff cté . L s configurations suivant s d modification d signatur s d méthod s sont c ll s conc rné s t l s méthod s proxi s associé s :

– odification d l'ordr d s paramètr s : → RT m(T2 p2, T1 p1), on a :

```
RT m (T1 p1, T2 p2) {
  return m(p2, p1);
}
```

– Suppr ssion d paramètr s : → RT m(T2 p2), on a :

```
RT m (T1 p1, T2 p2) {
  return m(p2);
}
```

– Ajout d paramètr s : → RT m(T1 p1, T2 p2, T3 p3)
C d rni r cas st l plus difficil à trait r, puisqu'un val ur initial n p ut pas toujours êtr fourni pour l s paramètr s ajoutés. En ff t, pour l s paramètr s comm un mot d pass , un cod s cr t ou un cod PIN, si c typ d paramètr st ajouté, il st difficil d prédir la val ur par défaut à aff ct r. ais dans c rtains cas où l'initialisation st possibl , nous pouvons avoir un méthod proxy comm c ll -ci :

```
RT m (T1 P1, T2 P2) {
  T3 p3 = null;
  return m(p1, p2, p3);
}
```

7. CONCLUSION

Dans c t articl , nous prés ntons l fram work Emb dDSU, un t chniqu d mis à jour dynamiqu d cod dans l cont xt d cart à puc Java. Nous prés ntons son archit ctur , son implém ntation, l s b nchmarks t l s travaux futurs. Emb dDSU st un fram work d HotSwUp basé sur la modification d la machin virtu ll Simpl RTJ, t st subdivisé n plusi urs parti s : off-card t on-card. Après implém ntation, l s b nchmarks prés ntés ont été réalisés sur un post d travail.
C p ndant, afin d'avoir d s b nchmarks plus réalist , actu ll m nt, d s travaux d transf rts d la machin virtu ll sur un board d'évaluation AT91 EB40 [16] sont n cours. L'idé étant d'approfondir l s b nchmarks pour obt nir l s t mps d' xécution à d s niv aux plus fin, t ls qu l t mps d transf rt du fichi r d DIFF, l t mps d'int rprétation du fichi r d DIFF, l s coût n consommation él ctriqu t surtout l s quantités d mémoir s consommé s tant au niv au d la RA qu'au niv au d l'EEPR .

Bibliographie

1. Agn s C. Noubissi, J an-Louis Lan t t Juli n Iguchi-Cartigny – Incr m ntal Dynamic Updat For Java Smart Cards Applications – IC NS'10, Franc , April 2010.
2. Agn s C. Noubissi, Juli n Cartigny t J an-Louis Lan t – Hot Updat s for Java Bas d Smart Cards – Third Workshop on Hot Topics in Softwar Upgrad s HotSwUp'11, Hannov r - G rmany, April 2011
3. Agnès C. Noubissi, Juli n Iguchi-Cartigny t J an-Louis Lan t – Cart à puc , v rs un duré d vi infini – aj cStic, Avignon, 2009.

4. Jonathan T. oor , icha l Hicks t Scott N ttl s – Dynamic softwar Updating, Programming Languag D sign and Impl m ntation – PLDI, AC , 2001.
5. S miconductors Austria GmbH Styria – http ://www.mifar .n t/.
6. Zhiqun Ch n – Java Card T chnology for Smart Cards – Addison, W sl y, 2000
7. Th Java Card 3.0 sp cification : http ://java.sun.com/javacard/
8. ilan Fort – Smart card application d v lopm nt using Java Card T chnology – S W S 2006.
9. Suriya Subramanian, icha l Hicks t Kathryn S. cKinl y – Dynamic Softwar Updat s : A V -C ntric Approach – PLDI, Jun 2009.
10. J sp r And rsson t Tobias Rit au – Dynamic d ploym nt of Java applications, Java for Emb dd d Syst ms Workshop – London, ay 2000.
11. Earl Barr, J. Frit Barn s, J ff Gragg, Raju Pand y t Scott alabarba – Runtim support for typ -saf dynamic java class s – EC P, 2000.
12. Al ssandro rso, Anup Rao, t ary J an Harrold – A t chniqu for dynamic updating of Java Softwar – ICS , 2002.
13. Karst n Nohl, David Evans, Starbug t H nryk Pltâ – R v rs -Engin ring a Cryptographic RFID Tag – USENIX S curity Symposium, San Jos , CA. July 2008.
14. Karst n Nohl t H nryk Plt – IFARE, Littl S curity, D spit bscurity – Pr s ntation on th 24th Congr ss of th Chaos Comput r Club in B rlin, D c mb r 2007.
15. G. d Koning Gans, Jaap-H nk Ho pman t Flavio D. Garcia – A FlR-i6(a)4.81944(.733( )-2.79909(t)-423.449(F)435(P)1.75