ISSN - 2170- 0656

CERISTNEWS

Bulletin d'information trimestriel

Onzième numéro - Décembre 2012

DOSSIER

Les Architectures de Procédés Logiciels : Etat de l'art

CENTRE DE RECHERCHE
SUR L'INFORMATION
SCIENTIFIQUE ET TECHNIQUE

cerist


ذكرى 50 للاستقلال
1962
2012
عيد الجزائر
2012 1962
50
قد كنا أمس عمالقة ★★★ في الحرب نذل أعادينا
وانا اليوم عمالقة ★★★ في السلم حماة مبادينا

Éditorial

M. Ahmed Nacer
Professeur à l'Université des Sciences et de la Technologie Houari Boumédiène (USTHB)

Notre monde d'aujourd'hui est un monde de logiciels, mais aussi un monde piloté par des logiciels car caractérisé par un fonctionnement de plus en plus automatisé à tous les niveaux de la vie sociale, économique, culturelle et autres.

Du domaine de l'informatique classique au domaine de l'informatique mobile dominée par une connectivité sans cesse croissante, l'on parle maintenant de l'internet des objets où les processus automatiques seront de plus en plus présents partout dans notre vie quotidienne.

La nécessité de mise en place de logiciels adéquats, qui a toujours été une préoccupation constante en génie logiciel, devient encore plus cruciale au vu des enjeux à venir. Aussi, le maître mot dans le développement de tout produit logiciel reste la « qualité », dans tout son sens, en termes de fiabilité, d'efficacité, de sûreté de fonctionnement, de maintenabilité et enfin d'ergonomie.

Les nombreuses expériences dans la production de logiciels complexes ont montré et prouvé que la qualité d'un produit logiciel passe nécessairement par la qualité du procédé ayant conduit à son développement. Plusieurs modèles de procédés logiciels (software process model) ont alors vu le jour intégrant le changement continu de méthodes et de pratiques de développement, suggérant de nouvelles méthodes et processus tout en s'adaptant à la progression rapide des technologies et des outils.

Dans cette quête permanente de la qualité des procédés logiciels et se basant sur la dualité produit logiciel / procédé logiciel, déjà sugérée par L. Osterweil en 1987 dans son célèbre article «Software processes are software too», B. Boehm relève que si les architectures logicielles sont efficaces pour la réutilisation des produits logiciels, elles sont aussi d'une réelle contribution pour la réutilisation des procédés logiciels.

Deux idées sont alors mises en exergue : l'exploitation d'une approche de réutilisation pour la modélisation des procédés logiciels, d'une part, et l'exploration du concept d'architectures logicielles pour la réutilisation des procédés logiciels, d'autre part.

C'est dans ce contexte que s'inscrit le thème central de ce dossier qui se veut une analyse des approches de réutilisation des procédés logiciels à base d'architectures logicielles, approches considérées comme solutions très prometteuses pour la modélisation de procédés logiciels de qualité.

Sommaire

Restitution des travaux de la commission nationale du haut et très haut débit

Créée en février2012, la commission nationale de large bande haut et très haut débit s'est réunie au CERIST le mardi 30 octobre 2012 sous la présidence de Moussa Benhamadi , ministre de la Poste et des Technologies de l'Information et de la Communication, en présence des responsables des opérateurs Mobilis, Algérie Télécoms, Arpt, et le CERIST.

M. Benhamadi a exhorté ces responsables et les personnels du secteur, à contribuer grandement à l'amélioration des services, en facilitant l'accès au haut et très haut débit. Les travaux de cette commission ont été lancés afin de trouver des solutions adéquates pour faire avancer les projets qui vont dans le sens de l'élargissement de l'utilisation des TIC, à commencer par les trois cycles scolaires : le primaire, le moyen et le secondaire ainsi que les universités et les centres de formation à l'échelle nationale.

M.Dabouz M'hand ,conseiller du ministre, a précisé que les propositions formulées tablent sur un débit au citoyen algérien de 2 Mbps à moyen terme « avec l'objectif d'atteindre le 8 et 10 Mbps au profit de 50% de la population d'ici 2015 »

Visite au CERIST des lauréats de la commune d'El Yachir Bordj Bou Arreridj

Le CERIST a reçu la visite d'une quarantaine de jeunes lauréats de la commune d'El Yachir de la ville de Bordj Bou Arreridj organisée par l'association de jeunes « EL Hidhab », le 27 décembre 2012.

Événements

Cette visite rentre dans le cadre du projet de promotion des Technologies de l'Information et de la Communication. Elle a commencé au niveau du bloc pédagogique où les élèves ont pu découvrir les salles intelligentes. Une présentation sur les activités du centre leur a été donnée par Mme Bebbouchi, chef de service de la valorisation des résultats de la recherche. Durant cette séance, les élèves ont, ainsi, pu tester le tableau blanc interactif. Par la suite ils ont visité la bibliothèque, puis se sont déplacés vers le bloc D où une présentation sur les réseaux de capteurs sans fil a été effectuée par l'équipe de recherche « Réseaux de capteurs et leurs applications ». La visite s'est achevée avec des prises de photos souvenir.

Festivités du 50ème anniversaire de l'indépendance : Lancement de la nouvelle version du site portail du CERIST

La cérémonie de remise des prix aux lauréats des concours organisés dans le cadre des festivités du cinquantième anniversaire de l'indépendance a eu lieu le lundi 31 décembre 2012 au sein du CERIST accompagnée par le lancement de la nouvelle version du portail du CERIST.

Dans le cadre de la célébration du cinquantième anniversaire de l'indépendance, deux concours ont été organisés:

- Un concours sur le relookage du site portail du CERIST en direction du personnel du centre auquel deux équipes ont pris part. Ces équipes étaient composées essentiellement d'informaticiens et de documentalistes.

Une commission ad-hoc avait été installée afin d'évaluer les deux propositions en tenant compte de plusieurs critères tels que l'architecture proposée, l'ergonomie, l'agencement du contenu, etc.

- Lauréats du concours « relookage du site portail du CERIST »
- Melle Tassadit Alloune
- Melle Amel Elaihar
- Mme Hadjira Derridj
- Melle Hayet Hadjar

Avec la participation de M.Bachir Djouabi pour l'aspect infographie et Melle Sabrina Benrahmoune pour l'aspect traduction.

- Un second concours, sponsorisé par l'opérateur de téléphonie mobile MOBILIS, concernait la réalisation d'un film documentaire scientifique en court métrage ouvert pour des jeunes élèves et étudiants de12 à 18 ans. Les participants à ce concours devaient réaliser une vidéo de 10 minutes maximum à l'aide d'une petite caméra d'un téléphone portable sur des thèmes scientifiques variés. Le but étant de montrer comment le média audiovisuel contribue au transfert des connaissances. La vidéo gagnante concerne le « Don sanguin et groupage » réalisée par deux jeunes sœurs : Melle Boucenna Maroua âgée de 16 ans et élève en deuxième année secondaire et Melle Boucenna Sarah âgée de 13 ans et élève en 3ème année moyenne. Les lauréates se sont vues attribuer deux téléphones portables multimédia offerts par MOBILIS.

Événements

11ème colloque africain sur la recherche en informatique et en mathématique appliquées CARI'2012

Le onzième Colloque Africain sur la Recherche en Informatique et en Mathématiques Appliquées (CARI'2012), s'est tenu du 13 au 16 octobre 2012 au siège du CERIST. La rencontre a rassemblé des représentants des universités africaines, des représentants de la communauté scientifique africaine, ainsi que des centres de recherche français et organismes internationaux.

CARI est devenu au fil des années un lieu privilégié de rencontres, de réflexion et d'échanges entre chercheurs et décideurs de haut niveau sur les problématiques des Sciences et Technologies de l'Information et de la Communication (STIC). Cette édition était organisée autour de six thématiques :

- Modélisation des Systèmes Complexes ;
- Signal, image, parole et multimédia ;
- Calcul Scientifique et Parallélisme ;
- Intelligence artificielle et Environnements collaboratifs ;
- Systèmes distribués, systèmes embarqués, réseaux, mobilité ;
- Génie logiciel : modèles, méthodes et applications.

Comme de coutume, le CARI'2012 a été précédé par des formations avancées du 9 au 11 octobre 2012 et a inclus des conférences invitées lors des sessions plénières. Quant aux formations avancées, elles étaient destinées aux doctorants et aux praticiens et accompagnées de travaux pratiques. Elles ont traité des thèmes suivants:

- Bioinformatique,
- Sécurité informatique,
- Systèmes d'information avancées,
- Méthodes numériques et modélisation.

INTERNATIONAL AUTUMN SCHOOL ON CYBER-PHYSICAL SYSTEMS

September 30 - October 03 2013

RESEARCH CENTER ON SCIENTIFIC AND TECHNICAL INFORMATION CERIST - ALGIERS, ALGERIA

ORGANIZED BY

CENTRE DE RECHERCHE SUR L'INFORMATION SCIENTIFIQUE ET TECHNIQUE | cerist

IMPORTANT DATES

- APPLICATION DEADLINE: 10TH OF SEPTEMBER
- REGISTRATION DEADLINE: 15TH OF SEPTEMBER

http://cpsschool2013.cerist.dz
ecolecps2013@cerist.dz

TOPICS

- Cyber Physical Systems
- Reliable Software
- Wireless Sensor Networks
- Security

SCIENTIFIC BOARD

- Pr. Nadjib Badache (CERIST, Algeria)
- Pr. Saddek Bensalem (University of Grenoble, France)
- Dr. Omar Nouali (CERIST, Algeria)
- Dr. Djamel Tandjaoui (CERIST, Algeria)

ORGANISING COMMITTEE:

Dalila Bebbouchi, Khedidja Bennadji, Saddek Bensalem, Fatiha Djetten, Yacine Mentalechta, Omar Nouali, Djamel Tandjaoui, Houria Zaidi.

INVITED LECTURERS

- Pr. Gilles Barthe - University of Madrid, (Spain)
- Pr. Sanjoy Baruah - University of North Carolina, (USA)
- Pr. Saddek Bensalem - University of Grenoble, (France)
- Dr. klaus Havelund - NASA, JPL, (USA)
- Pr. Yassine Lakhnech - University of Grenoble, (France)
- Pr. Jean Louis Lanet - University of Limoges, (France)
- Pr. Insup Lee - University of Pennsylvania, (USA)
- Dr. Axel Legay - INRIA, Rennes, (France)
- Dr. Susanne Lesecq - CEA-LETI, Grenoble, (France)
- Dr. François Pacull - CEA-LETI, Grenoble, (France)
- Pr. Radu Grosu - Technical University of Vienna, (Austria)
- Dr. Marco Roveri - FBK, Trento, (Italy)
- Dr. Harald Ruess - Fortiss, Managing Director, (Germany)

LE DOSSIER

Document spécial de 27 pages : 11/39

Les Architectures de Procédés Logiciels : Etat de l'art

Un dossier élaboré par :

Pr. Mohamed Ahmed-Nacer

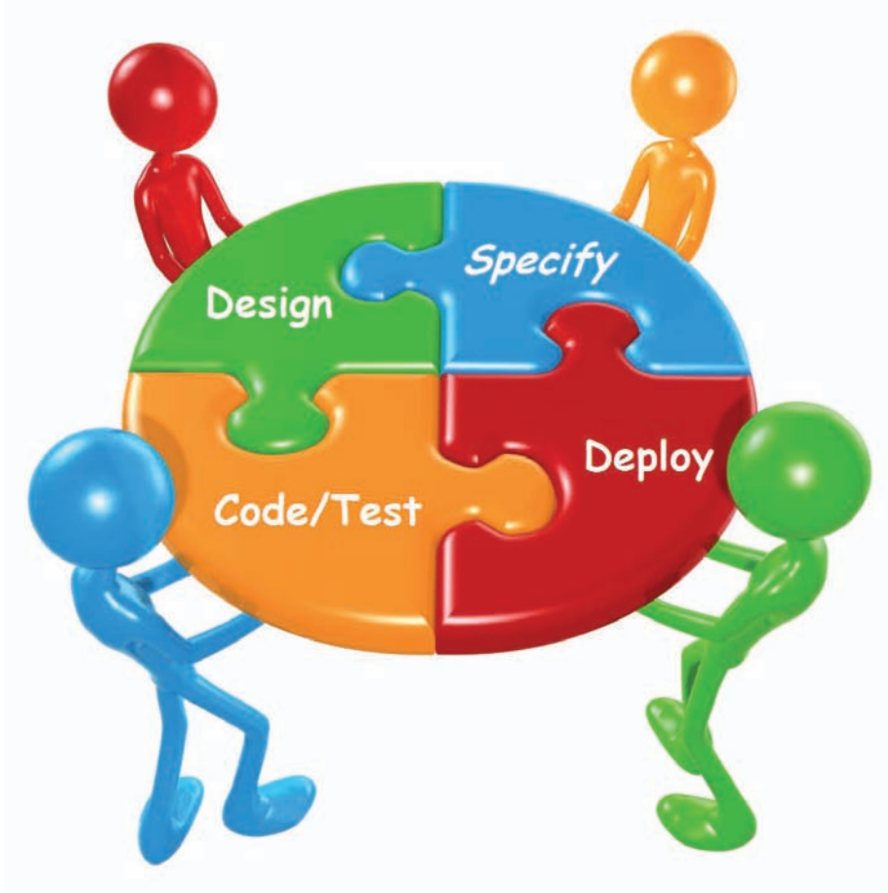
Université des Sciences et de la Technologie Houari Boumediène

Dr. Fadila Aoussat

Université des Sciences et de la Technologie Houari Boumediène

Pr. Mourad Oussalah

Université de Nantes



1. Introduction

Dans le domaine du génie logiciel, il est reconnu depuis longtemps qu'une application logicielle complexe est considérée comme un produit manufacturé complexe dont la réalisation doit s'intégrer dans une démarche méthodologique. Cette démarche méthodologique, qui a pour but la maîtrise de la complexité croissante des projets de développement de logiciels, est explicitée à travers l'utilisation de modèles de procédés logiciels (MPLs).

Un modèle de procédés logiciels correspond à la description d'un enchainement d'activités, de ressources, d'outils utilisés, ainsi que de la description des intervenants et de leurs rôles pour la réalisation puis la maintenance d'un produit logiciel. De ce fait, la qualité d'un modèle de procédé logiciel et la qualité d'exécution de ses procédés ont un impact direct sur la qualité du produit logiciel manufacturé.

Malheureusement, les modèles de procédés logiciels proposés ne sont pas toujours appropriés, car souvent trop complexes à modéliser [Starke (1994)] ; les concepts représentés dans ces modèles peinent à combiner des caractéristiques indispensables aux procédés logiciels (PLs) telles que la flexibilité et le contrôle d'exécution [Turk et al. (2000), Bendraou et al. (2008)], ou la simplicité et l'efficacité. C'est la raison pour laquelle ces modèles de procédés échouent, souvent, à s'imposer comme solution incontournable dans le domaine industriel.

De plus, la progression rapide des technologies et des outils de développement, le changement continu des méthodes et des pratiques de développement (développement orienté composant, programmation orientée aspect, programmation en binôme,...etc.), suggèrent de nouvelles méthodes et processus de développement, à l'instar d'UP, XP, Scrum, etc. Les procédés logiciels doivent alors intégrer continuellement ces nouvelles pratiques et s'adapter à ces nouvelles technologies.

Comme les procédés logiciels sont centrés humain [Sommerville et Rodden (1996)], l'expérience et le savoir-faire humain restent primordiaux lors de leur modélisation et de leur exécution. Ceci suggère alors l'exploitation d'une approche de réutilisation pour la modélisation des procédés logiciels.

De même, comme la répétitivité des tâches, l'importance de leurs interactions ainsi que l'exploitation des structures récurrentes font partie des caractéristiques intrinsèques du procédé logiciel, et que ces mêmes caractéristiques évoquent celles des architectures logicielles (ALs), ceci nous oriente alors naturellement vers l'exploration des architectures logicielles pour la réutilisation des procédés logiciels.

Cet aspect a déjà été relevé dans [Boehm (1995)] mettant en évidence la dualité produit logiciel / procédé logiciel concernant les architectures logicielles. En se basant sur l'article «Software Processes Are Software Too» [Osterweil (1987)], Boehm confirme que si les architectures logicielles sont efficaces pour la réutilisation des produits logiciels, elles sont aussi d'une réelle contribution pour la réutilisation des

procédés logiciels. «If open architectures are good for software product reuse, then their process counterparts will be good for software process reuse».

Comme le domaine des architectures logicielles est arrivé à un degré de maturité considérable, il est judicieux de tirer avantages des opportunités offertes par ce domaine pour promouvoir la réutilisation des procédés logiciels. En effet, l'exploitation des concepts architecturaux tels que les connecteurs, les styles architecturaux et l'utilisation de langages tels que les ADLs (Architecture Description Languages), peuvent être des atouts majeurs à la modélisation des procédés logiciels par approche de réutilisation.

Ce dossier porte sur une analyse des approches de réutilisation des procédés logiciels à base d'architectures logicielles.

La deuxième section est dédiée à la présentation des concepts de base liés aux procédés logiciels et aux architectures logicielles. La deuxième section discute des principales approches de réutilisation de procédés logiciels à base d'architectures logicielles. La quatrième section définit un cadre de comparaison où nous identifions les caractéristiques essentielles des procédés logiciels et les spécificités des approches de leur réutilisation à base d'architectures logicielles. La cinquième section présente une analyse des approches étudiées, une synthèse et un bilan sur l'état du domaine de la réutilisation des procédés logiciels à base d'architectures logicielles.

2. Concepts de base

2.1 Les modèles de procédés logiciels

Un modèle de procédés logiciels, ou modèle de PLs, est la représentation plus ou moins formalisée d'un ensemble de PLs. Il explicite les propriétés et les contraintes qui régissent le monde réel du développement en prenant en charge, entre autres, l'évolution prévue et imprévue [Kaba A.B. (1994)] de la réalité du développement

Différents modèles de PLs peuvent décrire différents angles de développement logiciel [Acuna et al. (2000)], se focalisant sur un aspect particulier du développement et reléguant d'autres aspects au second plan. Ainsi, différents langages de modélisation de PLs existent, suivant les buts recherchés de simulation, d'exécution, etc. Plusieurs classifications de modèles de PLs ont alors été définies, relativement aux critères suivants :

- Le niveau de formalisation du modèle de PL : non formel, semi formel, exécutable [Sanlaville (2005)].
- La nature centrale du modèle de PL : centrée activité, centrée rôle, centrée artefact, etc. [Hug (2009)].
- L'orientation du modèle de PL : orientée gestion de configuration, orientée décision, orientée stratégie...[Hug (2009)].

• Le type du langage de modélisation de PL: orienté objet, réseau de pétri, à base de règles... [Acuna et al., Zamli (2004), Bendraou et al. (2007)].

• Le type d'exécution supporté : exécution distribuée, simulation, etc. [Acuna et al. Zamli et Lee (2001), Bendraou et al. (2007)].

Les concepts des modèles de PLs varient selon les orientations adoptées. Il existe autant de métamodèles que d'orientations de PLs [Hug (2009)]. Cependant, tous les PLs sont conformes au même noyau conceptuel (figure 1) : Le PL est un enchainement d'activités, chacune nécessitant des produits «produit de travail» en entrée (inputs) pour fournir des produits en sortie (outputs). Le PL étant centré humain, une activité est sous la responsabilité d'un rôle «Role».

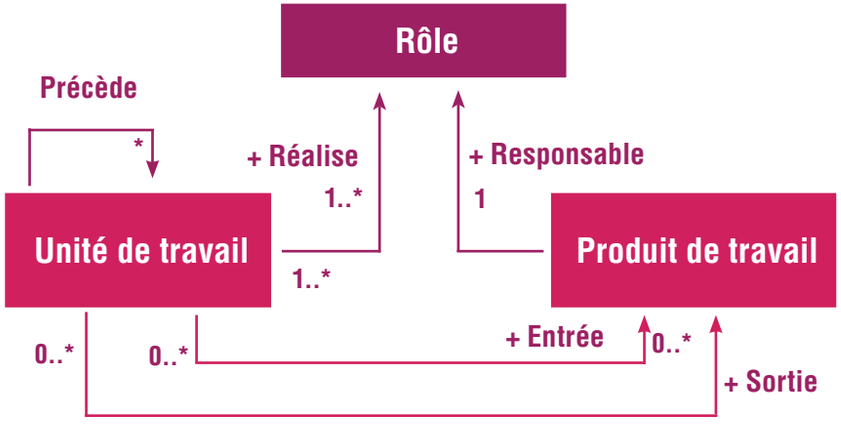


FIG. 1 : Le Métamodèle noyau d'un procédé logiciel.

Ainsi, les concepts de base de tout modèle de PL sont : Activité, Produit et Rôle.

Il est clair que d'autres concepts tels que stratégie, organisation ou outil peuvent être présents dans un métamodèle de PL, dépendant de l'orientation et de la spécialisation des modèles de PLs que l'on veut décrire.

2.2 Les architectures logicielles

L'accroissement de la complexité des systèmes logiciels à développer en taille, architecture, standardisation, intégration, et autres, a obligé les développeurs à reconsidérer leur vision du logiciel. Le besoin d'abstraire et d'architecturer les logiciels avant de les développer est alors devenu indispensable. C'est dans ce contexte que les architectures logicielles, ou ALs, se sont imposées comme solution à la maîtrise de la complexité des logiciels.

Même si la plupart des définitions de l'architecture logicielle convergent, aucune ne s'impose comme référence. La définition des ALs a évolué avec les avancées des travaux dans ce domaine [Perry et Wolf (1992), Garlan (1995), Bass et al. (1998), Garlan (2000), Szyperski (2002), Fielding (2000), Leymonerie (2004)].

Plus précisément, nous considérons une architecture logicielle comme un ensemble d'unités de traitements ou de stockage (composants),

qui interagissent à travers des unités d'interactions (connecteurs). Cet assemblage (configuration) est régi par des règles et des contraintes de composition qui prennent en compte les propriétés des éléments architecturaux entrant dans la configuration. Ainsi, les éléments architecturaux de base de toute architecture logicielle sont : composant, connecteur et configuration.

Les architectures de PLs sont aussi des modèles de PLs mais décrits en tant qu'architectures, en exploitant les concepts architecturaux. Ainsi, il est naturel de retrouver des concepts tels que le composant procédé, le connecteur procédé ou la configuration procédé dans l'architecture d'un PL.

3. Les approches de réutilisation de procédés logiciels à base d'architectures logicielles

Plusieurs approches de réutilisation des PLs à base d'ALs ont été proposées (figure 2). Chaque approche apporte une solution particulière pour répondre à une préoccupation particulière (distribution, évolution, hétérogénéité, simulation, dynamicité, etc.). Néanmoins, ces approches ont toutes un point en commun ; elles exploitent des concepts architecturaux pour assoir leur solution.

Selon l'élément architectural sur lequel se focalisent ces approches de réutilisation, celles ci sont classées en trois catégories :

- Des approches orientées composants,
- Des approches orientées connecteurs,
- Des approches orientées configurations.

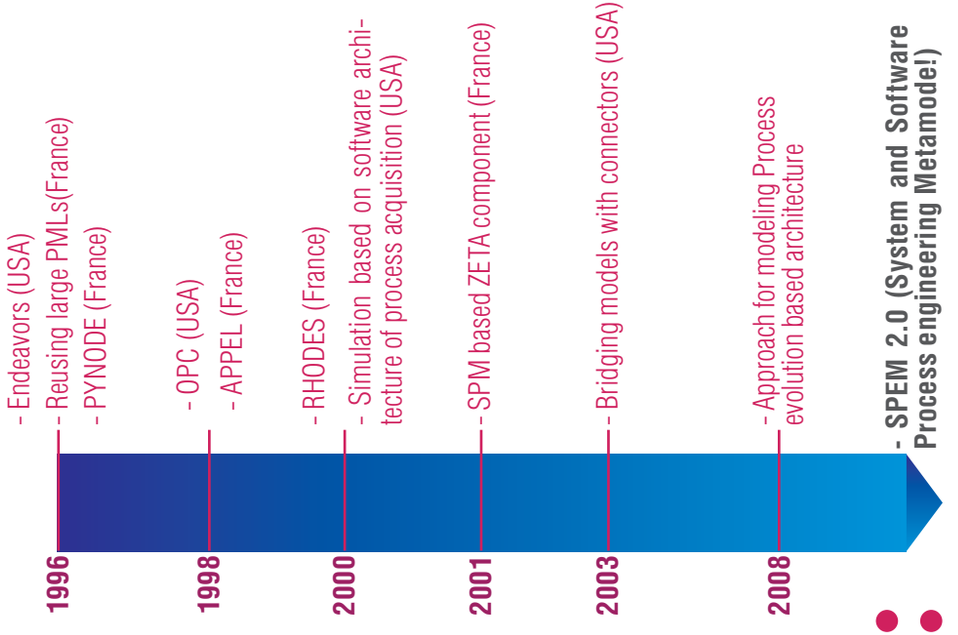


Figure 2 : Approches étudiées de réutilisation de PLs à based'ALs.

Le tableau 1 ci dessous résume ces approches et avance leurs objectifs et leurs points forts.

Approche	Objectifs	Points forts	Formalisme de modélisation
Endeavors [Hitomi et al. (1997)]	Traitement des procédés WorkFlow, spécifique au travail distribué et dynamique.	– Distribution via le web. – Indépendant des plateformes d'exécution. – Représentation graphique du modèle de PL.	OO
PYNODE [Avrilionis et al. (1996)]	Réutilisation dynamique des composants hétérogènes	– Exécution dynamique. – Hétérogénéité des fragments PLs.	OO
PLs à base de gestion de configuration [Belkhatir et Estublier (1996)]	Réutilisation des PLs par versionnement des composants	– Réutilisation par versionnement. – Contrôle de la cohérence du résultat	OO
OPC [Gary et al. (1998)]	Réutilisation dynamique de composants hétérogènes	– Exécution dynamique. – Hétérogénéité des fragments PLs	OO
APEL [Dami et al. (1997)]	Exécution distribuée de PL hétérogènes indépendamment des moteurs d'exécution.	– Exécution distribuée – Indépendant des moteurs d'exécution – Hétérogèneité des modèles de PLs.	OO
RHODES [Coulette et al. (2000)]	Réutilisation des connaissances de développement, assistance à la modélisation de PLs.	– Assistance à la modélisation. – Vérification de la cohérence du modèle.	OO
App. d'assemblage de modèles basée connecteurs [Med-vidovic et al. (2003)]	Définition de différents connecteurs de modèles de données des différentes phases de développement.	– Identification de règles, pour le passage d'un modèle de données à un autre. – Identification des propriétés communes entre les connecteurs PLs	ADL
ADL ZETA pour la modélisation des interactions PLs [Alloui et Oquendo (2001)]	Prise en charge des différentes interactions des modèles de PLs	Définition de modèles d'interactions et de modèles d'interactions types.	ADL
Simulation d'architectures de PLs d'acquisition [Choi et Scacchi (2001)]	Simulation, exécution concurrente et distribuée des modèles PLs d'acquisition	Simulation de l'exécution de modèles de PLs d'acquisition (procédés complexes).	ADL
Modélisation de PL évolutifs [Dai et al. (2008)]	Réutilisation des composants PL évolutifs	Définition d'un langage pour la description des architectures de PLs	ADL
SPEM [Object Management Group (2008)]	Réutilisation de composants PLs.	Définition de métamodèles pour la réutilisation des PLs à base de composants PLs	OO

Tableau. 1 : Résumé des approches étudiées de réutilisation de PLs à base d'ALs.

Il peut être constaté que ces approches n'offrent pas de solution générique ; les solutions proposées traitent de problèmes particuliers et proposent des solutions particulières, ce qui limite la réutilisation des PLs.

On peut également constater que les approches les plus anciennes sont influencées par la communauté industrielle ; ainsi, les architectures de PLs sont décrites à l'aide de langages orientés objets (OO). Par contre, les approches les plus récentes sont plutôt influencées par la communauté académique ; de ce fait, les architectures de PLs sont décrites en exploitant des langages de description d'architectures (ADLs : Architecture Description Language).

4. Cadre de comparaison pour les solutions de réutilisation de modèles de procédés logiciels à base d'architectures logicielles

Plusieurs cadres de comparaison ont été proposés dans le domaine des PLs, parmi ceux ci Bendraou et Gervais (2007), [Zamli et Lee (2001), Acuna et al. (2000), Ambriola et al. (1997)]. Cependant, l'introduction de nouveaux paradigmes dans la modélisation et l'exécution des PLs tels que la programmation orientée aspects [Mishali et Katz (2006)], les systèmes multi-agents [Wang (2002)] et les ALs [Choi et Scacchi (2001)], entre autres, mène souvent à l'introduction de nouveaux critères de comparaison et impose dans certains cas la définition de nouveaux cadres de comparaison spécifiques à ces nouveaux paradigmes.

Comme les solutions de modélisation de PLs à base d'ALs ont pour particularité d'être à l'intersection de deux axes de recherche que sont l'ingénierie des PLs et l'ingénierie des AIs, il est alors nécessaire de définir un cadre de comparaison approprié tenant compte de l'introduction des ALs comme paradigme de modélisation de PLs .

Comme déjà souligné, les architectures de PLs sont avant tout des modèles de PLs ; de ce fait, elles ont les mêmes caractéristiques et objectifs que les modèles de PLs traditionnels. Par ailleurs, les PLs sont modélisés en exploitant les concepts des ALs (composant, connecteur, configuration, etc.) ; ils sont alors considérés comme des AIs, ils ont donc les mêmes caractéristiques et objectifs que les ALs.

Notre cadre de comparaison doit prendre en compte au moins ces deux axes : Axe des PLs et axe des ALs. Ces deux axes définissent l'aspect technique des solutions de réutilisation de PLs à base d'AIs ; ils se focalisent sur le côté quantitatif de l'architecture de PL (Figure 3). L'aspect qualitatif de l'architecture de PL doit aussi être évalué ; ce qui constitue le troisième axe de notre cadre de comparaison. Ainsi, nous définissons les critères d'évaluation selon les axes suivants :

• 1er axe - Les critères d'évaluation selon l'axe PL : ces critères permettent d'évaluer les concepts et langages de PLs utilisés pour décrire le PL indépendamment de la vision architecture.

• 2ème axe - Les critères d'évaluation selon l'axe AL : contrairement aux caractéristiques précédentes, ces critères permettent d'évaluer les concepts et langages exploités pour décrire la structure abstraite de l'architecture de PL indépendamment de l'aspect procédé.

• 3ème axe - Les critères d'évaluation qualitative : ces critères permettent d'évaluer la qualité des modèles de PLs à base d'ALs.

Nous détaillons ces critères dans les sections suivantes.

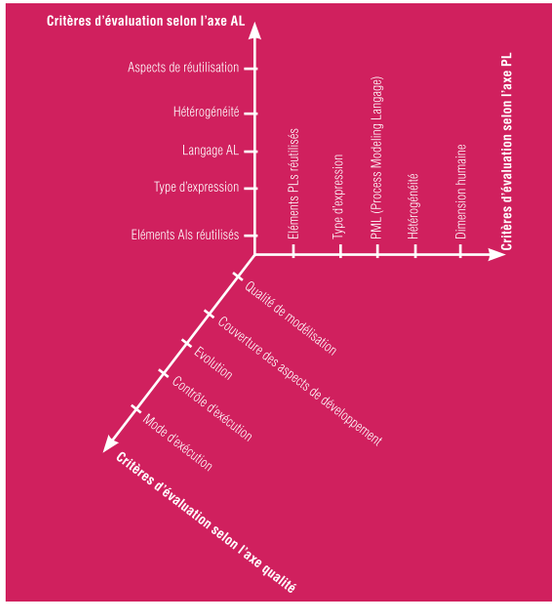


Figure 3 :

Les axes d'évaluation des approches de réutilisation de PLs à base d'ALs.

4.1 Les critères d'évaluation selon l'axe procédé logiciel

Ces critères définissent les caractéristiques des modèles de PLs indépendamment de la vision AL ; il est alors légitime de prendre en compte les classifications et les cadres de comparaison déjà établis dans ce domaine.

Éléments PLs réutilisés : - De base (activité, Rôle et Produit) - Secondaire (ressource, acteur,…)
Types d'expression : - Non formel - Semi formel - Formel
PMLs (Process Modeling Languages) : - Orienté objet - Réseau de Pétri - A base de règles - Fonctionnel
Hétérogénéité : - Syntaxique - Sémantique - Plateforme d'exécution - Homogène
Dimension humaine : - Assistance à l'exécution - Contrôle de l'interaction humaine

Tableau 2 – Les critères d'évaluation selon l'axe PL.

Le tableau 2 résume les critères sélectionnés pour évaluer l'aspect technique des solutions de réutilisation selon l'axe PL. Ces critères, inspirés des classifications de Longchamps [Acuna et al. (2000)], Conradi et al [Fuggetta (2000)], zamli [Zamli et Lee (2001), Zamli (2004)], correspondent aux significations suivantes :

Les éléments PLs réutilisés

Comme cité précédemment le noyau conceptuel du PL est constitué des concepts Activité (Unité de travail), Produit de travail et rôle. D'autres éléments dits secondaires peuvent être utilisés pour décrire d'autres facettes du PL. Ainsi, nous regroupons les éléments PLs réutilisés en deux groupes :

- Groupe de base : Les éléments «activité» «rôle» et «produit» sont considérés comme des concepts de base dans les PLs. Ces concepts restent les éléments de base pour les PLs à base d'ALs.
- Groupe secondaire : Ces éléments décrivent les préoccupations et les objectifs spécifiques à certains modèles de PLs, ces derniers pouvant être dédiés à la gestion de configuration, la gestion du temps ou des ressources de développement [Gallardo (2000)], etc.

Des concepts additionnels peuvent être intégrés pour prendre en compte ces préoccupations, les concepts «ressources», «outils», «stratégie» ou «guidance» représentent quelques exemples de concepts PLs secondaires.

Le type d'expression

Ce critère permet de spécifier le degré d'abstraction des formalismes utilisés lors de la modélisation des PLs à base d'ALs. Comme pour les PLs, selon l'objectif et la complexité du modèle de PL, nous distinguons plusieurs types d'expression. Dans notre cadre de comparaison, nous classons les modèles de PLs selon trois types de formalismes.

- Non formel : Les modèles de PLs sont décrits sans utiliser un formalisme strict mais plutôt des formalismes à base de langage naturel. L'objectif est la compréhension du PL.
- Semi formel : Les modèles de PLs sont décrits graphiquement, l'objectif est la structuration du PL.
- Formel : Les langages utilisés sont des langages qui ont une syntaxe et une sémantique rigoureuses, l'objectif est de permettre l'exploitation effective par exécution du modèle de PL.

Le PML (Process Modeling Language)

Un PML (Process Modeling Language) est un langage qui permet de modéliser des PLs. Un nombre important de PMLs ont été définis, exploitant différents paradigmes [Zamli et Lee (2001)]. Pour la comparaison des modèles de PLs à base de concepts architecturaux, nous utilisons les différentes catégories de formalismes définies dans la classification d'Ambriola et al [Fuggetta (2000)]. Ainsi, nous utilisons les catégories suivantes :

- PMLs Orientés Objet : le PML se base sur les concepts et les mécanismes orientés objets.

- PMLs à base de réseau de Pétri [Zamli et Lee (2001)].
- PMLs à base de règles.
- PMLs déclaratifs : le PML définit le modèle de PL comme un programme en exploitant les langages de programmation connus [Osterweil, L. (1987)].

L'hétérogénéité

L'hétérogénéité peut être évaluée principalement dans les modèles de PLs distribués. En effet, les différents fragments du modèle de PL peuvent être hétérogènes selon trois niveaux :

- Syntaxique : les fragments de modèles de PLs sont modélisés en exploitant différents PMLs.
- Sémantique : les fragments de modèles de PLs ne respectent pas le même métamodèle.
- Plateforme d'exécution : les fragments de PLs peuvent s'exécuter sur des plateformes d'exécution différentes.
- Homogène (pas d'hétérogénéité) : les fragments de modèles de PLs respectent la même syntaxe, la même sémantique et s'exécutent sur la même plate-forme d'exécution.

La dimension humaine

Les PLs sont orientés humain ; les tâches de réflexion, d'analyse et de création sont des tâches de développement typiquement humaine. L'influence des pratiques sociales et culturelles qui sont humaines et non techniques sur la modélisation et l'exécution du PL a été déjà mise en évidence dans [Sommerville et Rodden (1996)]. La prise en compte de cette dimension lors de la modélisation des architectures de PLs est primordiale. Pour cela, nous définissons deux critères :

- Assistance à l'exécution : l'approche doit permettre l'identification des tâches typiquement humaines et offrir l'assistance adéquate au réalisateur de cette tâche lors de l'exécution du modèle de PL.
- Contrôle de l'interaction humaine : l'interaction humaine peut déterminer la réussite du modèle de PL. Ce critère permet d'évaluer l'importance donnée au contrôle des interactions humaines.

4.2 Les critères d'évaluation selon l'axe architecture logicielle

Pour l'identification des critères d'évaluation selon l'axe AL, nous nous inspirons des Frameworks [Accord (2002), Medvidovic et Taylor (1997), Babar et al. (2004), Mehta et al. (2000)] mis en place dans le domaine des ALs.

La réutilisabilité est l'essence même des ALs, et par transition l'essence des PLs à base d'ALs. Ces modèles de PLs exploitent des concepts réutilisables tels que le composant, le connecteur ou la configuration.

Éléments ALs réutilisés : - Composant - Connecteur - Configuration - Style
Type d'expression : - Implicite - Prédéfini- Explicite
Langages d'AL : - Langage orienté objet - ADL spécifique aux PLs – ADL standard
Hétérogénéité : - Homogène - Hétérogène (composant, connecteur, configuration)
Aspect de réutilisation : - Mécanismes de réutilisation - Espace de stockage adopté - Portée de la réutilisation : Interne ou externe au système - Déploiement et génération de code

Tableau 3 – Les critères d'évaluation selon l'axe AL

Le tableau 3 résume les critères adoptés pour l'évaluation des approches de réutilisation de PLs selon l'axe AL. Ces critères correspondent aux significations suivantes :

Éléments ALs réutilisés

Une AL décrit le système comme un ensemble de composants (unités de calcul ou de stockage) qui communiquent entre eux par l'intermédiaire de connecteurs (unités d'interaction). La plupart des travaux existants considèrent les concepts «composant», «connecteur» et «configuration» comme concepts de base de toute AL. Ce constat reste valable pour les modèles de PLs à base d'ALs.

L'importance de ces concepts varie d'une approche de réutilisation de PLs à une autre. Cette variation dépend de l'interprétation que peut donner l'auteur aux entités réutilisables. En plus des éléments architecturaux de base à savoir : composant, connecteur et configuration, nous introduisons le concept «style». Ces concepts architecturaux gardent les mêmes définitions adoptées par la communauté des ALs, ainsi :

- Composant : est une entité qui fournit des fonctionnalités de calcul et de stockage spécifiques aux PLs. Le composant interagit avec les autres composants pour réaliser un ou plusieurs objectifs de l'architecture de PL.
- Connecteur : est un bloc de construction architectural utilisé pour modéliser les interactions entre les composants PLs et pour spécifier les règles qui régissent ces interactions.
- Configuration : représente un graphe de composants et de connecteurs et définit la façon dont ils sont reliés entre eux.

• Style : permet de capturer les caractéristiques des structures et des comportements récurrents des PLs.

Le type d'expression

Le type d'expression des éléments réutilisés peut être :

• Implicite : exploité mais de manière non formelle sans utiliser de langage de modélisation, ou bien sans avoir d'existence propre.

• Prédéfini : Les instances des éléments prédéfinis sont de structures et des fonctionnalités connues, stockées et réutilisées.

• Explicite : Les instances des éléments ont une syntaxe et une sémantique rigoureuse. Des instances peuvent être rajoutées ou modifiées.

Les langages d'ALs

Les ADLs (Architecture Description Langages) ou langages de description d'architectures sont des formalismes spécifiques aux ALs. Ils sont utilisés pour décrire la structure d'un système comme un assemblage d'éléments logiciels. Pour la modélisation des concepts architecturaux, nous considérons trois catégories :

• Les Langages orientées objets : certaines architectures sont décrites à l'aide de langages orientés objets et exploitent les mécanismes objet tels que l'instanciation, l'agrégation et l'héritage.

• ADLs spécifiques aux PLs : Ce sont des ADLs spécifiques à la description des architectures de PLs.

• ADLs standards: Ce sont des langages destinés à décrire des ALs mais qui sont non destinés au domaine des PLs.

L'hétérogénéité

Les modèles de PLs à base d'AL peuvent être :

• Hétérogènes : les composants ou les connecteurs de l'architecture de PL peuvent être hétérogènes syntaxiquement ou sémantiquement.

• Homogènes : le modèle de PL à base d'AL exploite les mêmes concepts et le même langage lors de la modélisation.

Les aspects de réutilisation

Ce critère permet d'évaluer certains aspects que nous jugeons importants pour faciliter la réutilisation des PLs. Ainsi, nous considérons :

• Les mécanismes utilisés pour la réutilisation effective tels que la composition et certains mécanismes orientés objets tels que l'héritage, l'instanciation, etc.

• L'espace de stockage et les stratégies de stockages adoptées : les espaces de stockage peuvent être des bases de données, des fichiers ou des ontologies, etc.

• Portée de la réutilisation [Oussalah (2005)] : nous identifions deux niveaux : la portée de réutilisation interne où les blocs sont réutilisés par le système qui les a créés seulement, et la portée de réutilisation externe où les blocs (composants, connecteurs) peuvent être exportés et réutilisés par d'autres systèmes [Oussalah (2005)].

• Le déploiement et la génération du modèle de PL : ce critère permet d'évaluer les mécanismes mis en place pour permettre la génération du modèle de PL final.

4.3 Les critères d'évaluation selon l'axe qualité

Afin d'évaluer la qualité des PLs à base d'ALs modélisés, nous identifions un certain nombre de critères de qualité. La plupart des critères de qualité sont communs aux modèles de PLs et aux ALs. En effet, la facilité de modélisation, la compréhension, ainsi que l'évolution sont des caractéristiques qui peuvent être attribuées aussi bien aux PLs qu'aux ALs. Les modes d'exécution possibles pour les architectures de PLs sont aussi recensés et s'inspirent des modes d'exécution des deux domaines. Nous rajoutons le critère « contrôle d'exécution » comme critère d'évaluation ; ce critère est spécifique aux PLs.

Le tableau 4 ci dessous résume les critères de qualité adoptés afin d'évaluer les PLs à base d'ALs.

Qualités de modélisation : - Facilité de modélisation - Compréhension - Cohérence et persistance
Mode d'exécution : - Simulation - Distribué - Hétérogène – Dynamique - Incrémental- Itératif - Standard
Contrôle de l'exécution
Évolution : - Statique - Dynamique

Tableau 4 – Les critères d'évaluation selon l'axe qualité.

Qualité de modélisation

Ce critère est relatif à la qualité de la modélisation. Cette qualité peut être évaluée par:

• La facilité de modélisation : la modélisation doit être facile et indépendante du langage de modélisation de PL utilisé. Des outils d'assistance et de guidance doivent être intégrés pour faciliter la modélisation du PL.

• La compréhension : le résultat de modélisation doit être facilement compréhensible.

• La cohérence et la persistance du résultat : la solution de réutilisation doit permettre la vérification du résultat de modélisation. Cette option est indispensable pour tous les PLs et particulièrement pour les architectures de PLs, car la modélisation de PLs à base d'ALs prend en compte deux aspects : l'aspect structurel et l'aspect contenu.

Mode d'exécution

Ce critère permet de spécifier les modes d'exécution possibles pour une architecture de PL. Nous constatons que la plupart des modes d'exécution des deux domaines (AL et PL) sont similaires. Aussi, afin de prendre en compte les modes d'exécution des processus agiles, nous introduisons les modes d'exécution itératifs et incrémentaux, considérés comme des types particuliers de l'exécution dynamique. Les modes d'exécution identifiés se résument comme suit :

• Simulation : la simulation de l'exécution d'un modèle de PL est une exécution virtuelle du modèle de PL. L'objectif de la simulation est d'étudier le comportement de certains types de modèles de PLs dont l'exécution réelle est confrontée à des difficultés tels que leur complexité, faisabilité, coût ou temps d'exécution.

• Distribué : l'exécution se fait sur des plateformes ou des environnements de travail distribués; l'intérêt est de permettre le travail d'équipe de développeurs sur des sites géographiquement distants.

• Hétérogène : l'exécution hétérogène est souvent associée à l'exécution distribuée. Cela signifie que les modèles de PLs s'exécutent sur des plateformes ou des moteurs d'exécution hétérogènes.

• Dynamique : Il est dans ce cas possible d'effectuer des modifications sans interrompre l'exécution du modèle de PL.

• Incrémental : C'est un type particulier d'exécution dynamique. Le PL est modélisé puis exécuté partie par partie (incrément). L'objectif de ce type d'exécution est d'avoir un résultat opérationnel le plutôt possible dans le cycle de développement.

• Itératif : C'est un autre type particulier d'exécution dynamique où certains enchainements du modèles de PLs sont repris et ré-exécutés.

• Standard : l'exécution est lancée une seule fois ; elle est standard et n'appartient à aucune des catégories précédentes.

Contrôle de l'exécution

Le contrôle d'exécution est un aspect important dans le modèle de PL. La solution proposée doit permettre un contrôle d'exécution du modèle de PL et gérer les adaptations effectuées.

Évolution

Les PLs comme les ALs sont par nature évolutifs. Ainsi, ce critère permet d'évaluer les mécanismes mis en place pour promouvoir les différents types d'évolution du PL. En général, deux types d'évolution sont possibles :

• Statique : L'évolution n'est possible que lorsque le PL n'est pas en exécution.

• Dynamique : L'évolution est possible lors de l'exécution du PL.

5. Évaluation des approches de modélisation et d'exécution de PLs à base d'AIs

Nous présentons, dans ce qui suit, les évaluations des approches de réutilisation suivant les trois axes (axe PL, axe AL et axe qualité).

5.1 Évaluation des approches de réutilisation de PLs à base d'ALs selon l'axe PL

Le tableau 5 résume les caractéristiques des approches de réutilisation de PLs étudiées selon l'axe PL.

<table>
<tr><th rowspan="2">Approche</th><th colspan="2">Éléments PLs réutilisés</th><th rowspan="2">T.E.</th><th rowspan="2">PML</th><th rowspan="2">Hétérogénéité</th><th colspan="2">Dimension humaine</th></tr>
<tr><th>Base</th><th>Additionnel</th><th>Assistance</th><th>C.I.</th></tr>
<tr><td>Endeavors [Hitomi et al. (1997)]</td><td>Activité, Produit</td><td>Ressource</td><td rowspan="5">Formel et Semi formel</td><td>ObjV, Lisp (OO)</td><td>Plate-forme d'exécution</td><td>Représentation graphique</td><td rowspan="5">C.I.</td></tr>
<tr><td>PYNODE [Avrilionis et al. (1996)]</td><td>Activité, Produit, Rôle</td><td>—</td><td>ObjV, Lisp (OO)</td><td>Syntaxique</td><td>Représentation graphique</td></tr>
<tr><td>PLs à base de gestion de configuration [Belkhatir et Estublier (1996)]</td><td>Activité, Produit</td><td>version</td><td rowspan="3">dépend du modèle</td><td>Hétérogène</td><td>Vérification de la cohérence</td></tr>
<tr><td>OPC [Gary et al. (1998)]</td><td>Activité, Produit, Rôle</td><td>—</td><td>Syntaxique</td><td>Représentation graphique</td></tr>
<tr><td>APEL [Dami et al. (1997)]</td><td>Activité, Produit</td><td>Agent</td><td>Syntaxique Sémantique</td><td>Représentation graphique</td></tr>
</table>

RHODES [Coulette et al. (2000)]	Activité, Produit, Rôle	Stratégie		PBOOL+ (OO)	Homogène	représentation graphique, Détection des incohérences	C.I.
App. d'assemblage de modèles basée connecteurs [Med-vidovic et al. (2003)]	Activité, Produit	Phase	Non formel	—	—	—	
ADL ZETA pour la modélisation des interactions PLs [Alloui et Oquendo (2001)]	Activité, Produit	—	Formel	Dépend du modèle	Syntaxique Sémantique	Représentation graphique	
Simulation d'architectures de PLs d'acquisition [Choi et Scacchi (2001)]	Dépend du modèle de PL					Représentation graphique	
Modélisation de PL évolutifs [Dai et al. (2008)]	Activité, Produit	—	Semi formel	réseau de pétri	Homogène	Représentation graphique	
SPEM [Object Management Group (2008)]	Activité, Produit Rôle	Dépend de l'environnement respectant SPEM					
Abréviations : T.E. : Type d'Expression, C.I. : Contrôle d'Interactions.							

Tableau 5 – Évaluation des approches de réutilisation de PLs à base d'ALs selon l'axe PL.

De ce qui précède, nous apportons les remarques suivantes :

- Les concepts des modèles de PLs sont généralement des concepts de base et se limitent pour la plupart aux concepts « activité » et « produit ». Les approches étudiées n'ont pas exploité de concepts additionnels afin d'introduire les concepts architecturaux, exceptée l'approche de Belkhatir et al [Belkhatir et Estublier (1996)] qui utilise le « versionnement » dans la réutilisation de ses composants.
- La plupart des approches offrent des solutions qui modélisent des PLs formels ; l'objectif est souvent de fournir des modèles de PLs exécutables. Ces modélisations formelles sont souvent combinées à des modélisations semi-formelles qui se traduisent par des représentations graphiques du modèle de PL. Les représentations graphiques ont pour objectif de fournir une aide aux développeurs et aux utilisateurs des modèles de PLs.
- Aucun langage de modélisation de PLs n'affiche de tendances particulières et aucun langage n'est mentionné comme langage facilitant la réutilisation des modèles de PLs, excepté le langage PBOOL+ de l'environnement RHODES [Coulette et al. (2000)] qui intègre la notion de composant procédé et de réutilisation de PLs.

Dans certaines solutions, le langage de modélisation de PLs n'a pas d'influence et les composants sont des boites noires comme dans l'approche de Alloui et Oquendo [Alloui et Oquendo (2001)] et l'approche de Medvidovic et al [Medvidovic et al. (2003)]. Dans certains cas, le langage de modélisation de PL n'est même pas mentionné comme dans l'approche de Choi et Scatcchi [Choi et Scacchi (2001)].

- Concernant l'interaction humaine, aucune assistance particulière n'est fournie. L'interaction humaine se fait de manière implicite et n'est contrôlée par aucun mécanisme spécifique ; ce qui peut être considéré comme un inconvénient particulièrement pour les modèles de PLs à forte interaction humaine, excepté l'environnement RHODES qui fournit une assistance sous forme de «sketchs» et de guidances textuelles [Crégut et Coulette (1997)].

- L'approche SPEM possède la particularité d'être un métamodèle standard ; ce qui lui confère la possibilité de couvrir la modélisation d'un large éventail de PLs.

5.2 Évaluation des approches de réutilisation de PLs à base d'ALs selon l'axe AL

Le tableau 6 résume les caractéristiques des approches de réutilisation de PLs étudiées selon l'axe AL.

Approche	Éléments ALs réutilisés				Langage AL	Aspects de réutilisation	
	Com.	Con.	Conf.	Style		Mécanismes	Espace de stockage
Endeavors [Hitomi et al. (1997)]	Explicite	Implicite	Implicite		Orienté objet	Instanciation, Composition, Héritage Multiple	fichiers ascii
PYNODE [Avrilionis et al. (1996)]					Orienté objet	Instanciation, Composition dynamique, Héritage	BD de composants
PLs à base de gestion de configurations [Belkhatir et Estublier (1996)]					PIL (orienté objet)	Instanciation, Composition, Héritage, Versionnement	DBMS base vvde composants
OPC [Gary et al. (1998)]					Orienté objet	Instanciation, Composition dynamique, Héritage	Base de composants
APEL [Dami et al. (1997)]					Orienté objet	—	Pas de stockage
RHODES [Coulette et al. (2000)]					PBOOL+ (orienté objet)	Instanciation, Composition, Héritage modulaire, sous typage	Base de composants gérée par le SGBD OO jasmin
Simulation d'architectures de PL d'acquisition [Choi et Scacchi (2001)]					HLA (ADL)	Composition	bibliothèque de modèles d'acquisition
ADL ZETA pour la modélisation des interactions PLs [Alloui et Oquendo (2001)]		Exp.	Exp.	I.T.	ZETA (ADL)	Instanciation	Pas de stockage
App. d'assemblage de modèles basés connecteurs [Medvidovic et al. (2003)]	Imp.	Explicite	Imp.	I	—	—	—
Modélisation de PLs évolutifs [Dai et al. (2008)]	Explicite		Exp.		EPCDL (ADL pour PL)	Composition	base de composants
SPEM [Object Management Group (2008)]		Imp.	Imp.		Orienté objet	Composition, instanciation	—

Abréviations : Com. : Composant, Con.: Connecteur, Conf. : Configuration, Imp.: Implicite, Exp. : Explicite, I.T. : Interaction Type, PIL : Process Interconnection Language HLA : Hight Level Architecture, EPCDL : Evolution Process Component Description Language.

Tableau 6 – Évaluation des approches de réutilisation de PLs à base d'ALs selon l'axe AL.

L'analyse des caractéristiques des solutions de réutilisation étudiées selon l'axe AL (Tableau 6) montre que :

- La brique de base pour la réutilisation des PLs est le composant. La plupart des approches définissent le composant PL comme une unité de travail ou un enchainement d'unité de travail. Ainsi, le composant PL peut être un fragment de PL ou un modèle de PL.
- Le composant PL est dans la plupart des cas explicite, excepté l'approche de Medvidovic et al [Medvidovic et al. (2003)] qui se focalise sur les connecteurs.
- Concernant le concept de connecteur, celui ci est implicite dans les approches les moins récentes ; il est considéré comme une transmission de données ou un flux d'exécution. Ces transmissions n'ont pas eu beaucoup d'attention ; leur prise en charge est intégrée dans le PL et dépend du langage et de la nature du PL. Par contre, les solutions les plus récentes donnent plus d'importance aux connecteurs qui sont prédéfinis ou explicites. Il est à noter, cependant, que les définitions adoptées sur les connecteurs ne peuvent être généralisées, car dépendant du langage de modélisation du PL et de l'interprétation de l'auteur ; ces connecteurs restent spécifiques et la portée de leur réutilisation reste limitée à l'environnement d'origine.
- Dans la plupart des solutions, la notion de configuration est implicite. Dans les anciennes approches, la configuration implicite se traduit par l'exploitation de la représentation graphique du modèle de PL mais sans traitement ou analyse particulière au niveau de sa structure. La configuration est devenue explicite avec l'utilisation des ADLs qui permettent de décrire les éléments d'une configuration et de spécifier les contraintes d'assemblage qui la régissent.
- La notion de style architectural n'est pas exploitée dans les approches étudiées, excepté l'approche de Alloui et Oquendo [Alloui et Oquendo (2001)] qui définit des connecteurs types (Interaction Type) et des configurations types (Container type) qui sont des éléments de base d'un style architectural. La description de ces éléments types est possible grâce à l'utilisation de L'ADL ZETA qui est un outil de l'ingénierie des ALs.
- La majorité des langages d'ALs sont orientés objet ; la réutilisation est réalisée en exploitant les opérations offertes par le paradigme objet (instanciation, héritage, héritage multiple, composition, etc.).

L'utilisation d'ADLs n'a fait son apparition que dans les approches les plus récentes, ce qui a permis à ces approches d'expliciter la totalité des éléments architecturaux.

- Bien que la plupart des solutions offrent des dépôts de composants (base de composants, librairies, etc.) pour permettre la réutilisation des composants PLs, le détail de leur organisation ou de leur classification n'est pas présenté.
- Aucune approche n'a exploré la possibilité d'exploiter une ontologie de domaine. Ce choix est justifié par l'orientation adoptée par

les solutions proposées. L'exploitation d'une ontologie de domaine exige une autre vision et d'autres mécanismes de réutilisation (connaissances procédé, inférence, raisonnement...), ce qui n'est pas l'objectif de ces solutions.

- Les approches étudiées implémentent des ALs homogènes ; elles utilisent le même langage pour tous les composants et les connecteurs, excepté l'approche de Medvidovic et al [Medvidovic et al. (2003)] où les connecteurs peuvent être décrits en langage de programmation ou en langage naturel.
- La portée de réutilisation de toutes les approches étudiées est limitée à la réutilisation interne ; les solutions proposées traitent les préoccupations internes, et ne se focalisent pas sur la généricité et la portabilité des éléments architecturaux.
- Les approches étudiées ne proposent pas le déploiement d'Als. La plupart des solutions sont uniquement centrées sur le concept composant, le concept configuration étant implicite, ce qui ne permet pas de proposer un déploiement d'architectures.

5.2.1 Autres interprétations des termes des concepts architecturaux

Certaines approches utilisent les termes architecture/ connecteur/ configuration/ familly (qui est utilisé pour désigner un style dans l'environnement ACME studio) pour définir ou désigner des éléments autres que les éléments architecturaux ; leurs interprétations dépendent des solutions proposées, à l'exemple des concepts suivants:

- Composant : Dans APEL [Dami et al. (1997)], le composant est un composant produit et non un composant procédé. Il représente une partie de l'environnement de modélisation et d'exécution d'un PL, l'objectif étant d'avoir des modélisations et des exécutions distribuées de plusieurs modèles de PLs locaux.
- Connecteur : APEL [Dami et al. (1997)] utilise le terme connecteur pour décrire les liens (Data flow et Control flow) entre les activités PLs alors que le PL n'est pas décrit en orienté composant.
- Architecture : Borsoi et al [Borsoi et Becerra (2008)] utilisent le terme architecture pour décrire la structure générale du PL non pas en termes de composants mais en termes de « phase » et « d'activité ».
- Familly : Belkhatir et al [Belkhatir et Estublier (1996)] définissent le concept familly (qui peut laisser penser aux styles architecturaux) pour décrire un process unit (un composant) et ses versions.
- SPEM [Object Management Group (2008)] utilise le terme «configuration» pour décrire l'ensemble des éléments - qui ne sont pas forcément des éléments architecturaux- permettant de décrire un PL, qui n'est également pas forcément une configuration.

5.3 Évaluation des approches de réutilisation de PLs à base d'ALs selon l'axe qualité

Le tableau 7 résume les caractéristiques des approches étudiées de réutilisation de PLs selon l'axe qualité.

L'analyse des caractéristiques qualitatives des approches étudiées montre que :

- Les solutions proposées se focalisent sur la modélisation de PLs exécutables. De plus, l'utilisation de la modélisation à base d'ALs a permis de mettre en place des modèles de PLs distribués, hétérogènes, et d'offrir des possibilités d'exécutions dynamiques. Les caractéristiques de distribution, d'hétérogénéité et d'évolution sont considérées comme indispensables aux PLs de qualité. Ces caractéristiques ont pu être fournies en exploitant les concepts d'ALs.
- Les solutions étudiées n'offrent pas d'exécution incrémentale ou itérative, d'où l'intérêt portée aux méthodes agiles (qui sont incrémentales et itératives).
- Le contrôle d'exécution est sous la responsabilité du gestionnaire de PL. A part les représentations graphiques utilisées lors du contrôle d'exécution, aucun mécanisme spécifique à l'exécution centrée humain n'est explicité, ce qui augmente la dépendance de la qualité d'exécution des capacités humaines.
- Les ALs contribuent à améliorer la compréhension de modèles de PLs et à augmenter la facilité de modélisation. En effet, la formalisation puis l'utilisation de la structure abstraite du modèle de PL, en plus de la séparation entre interaction et traitement, améliorent la lisibilité et, par conséquent, la compréhension et la facilité de modélisation du PL. Il est à noter que ces deux critères (compréhension et facilité de modélisation) sont soumis à d'autres influences telles que le type du langage de modélisation de PL utilisé.
- Les ALs n'influent pas sur le critère de couverture des aspects de développement car celui-ci dépend exclusivement du langage de modélisation de PL utilisé.
- La plupart des approches modélisant formellement des PLs implémentent des mécanismes d'évolution (dynamique ou statique). Cependant, ces mécanismes sont divers et varient d'une approche à une autre. De plus, les évolutions offertes se situent toutes au niveau contenu et non au niveau structure (vision architecturale du modèle de PL).

Approche	Qualité de la modélisation	Mode d'exécution	Contrôle d'exé.	Évolution	
				Statique	Dynamique
Endeavors [Hitomi et al. (1997)]	—	Dynamique Hétérogène Distribué	—	Mécanismes d'intégration flexibles pour l'évolution incrémental du support	Déclarations dynamiques, modifications et extensions des champs, états, variables et interfaces objets
PYNODE [Avrilionis et al. (1996)]	Protocole spécifique à la composition dynamique		—	—	—
PLs à base de gestion de configuration [Belkhatir et Estublier (1996)]	Contrôle de la compatibilité, consistance lors de la sélection des composants	Distribué	—	Versionnement	—
OPC [Gary et al. (1998)]	—	Dynamique Hétérogène Distribué	Diag. état transition	—	Composition dynamique Changement dans les diagrammes d'état transitions des composants
APEL [Dami et al. (1997)]	—	Dynamique Distribué	—	—	—
RHODES [Coulette et al. (2000)]	Détecter les incohérences, gérer l'indéterminisme	Standard	—	mécanisme de poly-morphisme et de sous typage	—
App. d'assemblage de modèles basée connecteurs [Med-vidovic et al. (2003)]	—	—	—	—	—
ADL ZETA pour la modélisation des interactions PLs [Alloui et Oquendo (2001)]	—	Standard	modèles d'intera-ctions	—	—
Simulation d'architectures de P. d'acquisition [Choi et Scacchi (2001)]	—	Simulation	—	—	—
Modélisation de PL évolutifs [Kaba A.B. (1994)]	—	Standard	réseau de pétri	—	intégré dans le composant : composants évolutions
SPEM [Object Management Group (2008)]	—	Non défini	orienté objet	classe Variability	—

Tableau 7 – Évaluation des approches de réutilisation de PLs à base d'ALs selon l'axe qualité.

5.4 Bilan et discussion

L'étude des approches selon l'axe PL montre qu'il n'y a pas de tendance particulière dans le domaine de la réutilisation des PLs à base d'ALs : les concepts manipulés sont souvent les concepts noyau du PL, et dans la plupart des cas, il n'y a pas de concepts spécifiques à la réutilisation des PLs. Aussi, les langages de modélisation de PLs sont des langages connus dans le domaine de l'ingénierie des PLs et non spécifiques à la réutilisation des PLs.

D'autre part, il est constaté que la dimension humaine, l'une des spécificités de base des PLs, n'a pas eu beaucoup d'attention ; l'aide aux développeurs et surtout le contrôle des interactions humaines n'ont pas été pris en charge de manière efficace.

De même, l'étude de ces approches selon l'axe AL montre que les concepts architecturaux tels qu'ils sont définis dans le domaine des ALs ne sont pas bien exploités pour la réutilisation des PLs. Les concepts connecteur, configuration et style sont mal ou dans certains cas, pas du tout exploités. L'interprétation des connecteurs reste liée au langage de modélisation du PL utilisé, ce qui ne permet pas d'avoir des connecteurs fortement réutilisables.

L'exploitation du concept configuration reste implicite en utilisant des représentations graphiques sans réflexion ni raisonnement à ce niveau lors de la phase de modélisation.

Les styles architecturaux de PLs n'ont pas encore fait leur apparition. Les solutions proposées restent intuitives et se basent plutôt sur un besoin de réutilisation d'un certain type de PL que sur la définition d'une méthodologie pertinente pour la réutilisation globale des PLs, d'où l'apparition du domaine des architectures de PLs pour répondre à ces insuffisances.

Il est à noter également qu'aucune approche n'offre de mécanismes de déploiement et de génération de code, ce qui limite l'efficacité de ces solutions. En revanche, coté qualitatif, l'apport des ALs à la modélisation des PLs est considérable : L'exploitation des ALs pour la modélisation des PLs contribue à la compréhension et facilite la modélisation des PLs. Aussi, les ALs offrent des mécanismes qui permettent de modéliser de manière plus facile des PLs distribués, hétérogènes et dynamiques.

6. Conclusion

La mise en service (modélisation et exécution) d'un modèle de procédés logiciels suscite autant de préoccupations que la mise en service d'un logiciel «Software processes are software too».

La modélisation et l'amélioration des procédés logiciels restent un sujet de recherche d'actualité. La concurrence technologique et commerciale entre développeurs de logiciels, les pressions du marché du logiciel (produire vite, bien et pas cher), en plus des avancées technologiques qui doivent continuellement être prises en compte, sont autant de motivations qui incitent à l'amélioration des modèles de procédés logiciels.

La maturité du domaine des PLs en termes d'expériences de développement et en termes de concepts et de paradigmes nous oriente naturellement vers l'adoption d'approches de réutilisation pour la modélisation de PLs. Le sujet traité dans ce dossier concerne la modélisation et l'exécution des PLs par approche de réutilisation, à travers une évaluation des approches de réutilisation de procédés logiciels à base d'architectures logicielles. Les approches les plus significatives qui ont apporté des solutions aussi bien diverses qu'intéressantes sont passées en revue.

Afin de cerner les caractéristiques de ces approches, un cadre de comparaison a été défini. Ce dernier s'inspire des cadres de comparaison déjà établis aussi bien dans le domaine des PLs que dans le domaine des ALs. Aussi, les critères de comparaison et d'évaluation sont définis selon trois axes :

- Un axe décrivant les aspects techniques des solutions de réutilisation selon la vue PL.
- Un axe décrivant les aspects techniques des solutions de réutilisation selon la vue AL.
- Un axe décrivant l'aspect qualitatif des solutions de réutilisation.

L'étude d'évaluation de ces approches de réutilisation montre que l'exploitation des ALs est considérée comme une solution très prometteuse pour la modélisation de PLs de qualité ; elle permet non seulement d'augmenter la réutilisabilité des modèles de PLs, mais aussi d'assurer d'autres critères de qualité tels que la compréhension et la facilité de modélisation. Cette approche de modélisation de procédés logiciels à base d'architectures logicielles peut également s'appliquer à différents procédés logiciels en utilisant différents concepts ou différents langages de modélisation, ce qui augmente son efficacité en tant que solution de réutilisation.

Néanmoins, il faut noter que dans les approches étudiées, l'exploitation des concepts architecturaux n'a pas été optimale. Certains concepts tels que les styles architecturaux ne sont pas exploités, et rares sont les solutions qui ont utilisé tous les éléments architecturaux en même temps ; de nouveaux travaux sont en cours pour pallier à ces insuffisances et proposer de nouvelles solutions plus pertinentes.

Références

- Accord, P. (2002). Etat de l'art sur les langages de description d'architecture (adls). http://www.infres.enst.fr/projets/accord/lot1/lot_1.1-2.pdf.

- Acuna, S.T., Antonio, A.D., Ferre, X., Lopez, M., Mate, L., Estero, S. (2000) The software process: Modeling, evaluation and improvement. in Handbook of Software Engineering and Knowledge Engineering, S.K. Chang, Editor. 2000, World Scientific Publishing Company: Singapore. p. 193-237.

- Alloui, I. et F. Oquendo (2001). Supporting decentralised software-intensive processes using zeta component-based architecture description language. In ICEIS (1), pp. 207–215.

- Ambriola, V., R. Conradi, et A. Fuggetta (1997). Assessing process-centered software engineering environments. ACM Trans. Softw. Eng. Methodol. 6(3), 283–328.

- Avrilionis, D., N. Belkhatir, et P.-Y. Cunin (1996). A unified framework for software process enactment and improvement. In 4th International Conference on the Software Process, Washington, DC, USA, pp. 102. IEEE Computer Society.

- Babar, M. A., L. Zhu, et R. Jeffery (2004). A framework for classifying and comparing software architecture evaluation methods. In Proceedings of the 2004 Australian Software Engineering Conference, ASWEC '04, Washington, DC, USA, pp. 309–. IEEE Computer Society.

- Bass, L., P. Clements, et R. Kazman (1998). Software architecture in practice. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc.

- Belkhatir, N. et J. Estublier (1996). Supporting reuse and configuration for large scale software process models. In 10th International Software Process Workshop, pp. 35. IEEE Computer Society. Washington, DC, USA

- Bendraou, R., M.-P. Gervais, X. Blanc, et J.-M. Jézéquel. (2008): "Vers l'Exécutabilité des Modèles de Procédés Logiciels", 14ème colloque international sur les Langages et Modèles à Objets (LMO'08), Montréal, Canada, pp. 155-170, (Revue des Nouvelles Technologies de l'Information (RNTI)), (ISBN: 978.2.85428.824.7) (2008)

- Bendraou, R. et M.-P. Gervais (2007). A framework for classifying and comparing process technology domains. In Proceedings of the International Conference on Software Engineering Advances, ICSEA '07, Washington, DC, USA, pp. 5–. IEEE Computer Society.

- Bendraou, R., A. Sadovykh, M.-P. Gervais, et X. Blanc (2007). Software process modeling and execution : The uml4spm to ws-bpel approach. In EUROMICRO-SEAA, pp. 314–321.

- Boehm, B. (1995). Software process architectures. In Software Architecture Workshop, 17th ICSE, ICSE'95.

- Borsoi, B. T. et J. L. R. Becerra (2008). A method to define an object oriented software process architecture.19th Australian Conference on Software Engineering, ASWEC 2008, 650–655.

- Choi, S. J. et W. Scacchi (2001). Modeling and simulating software acquisition process architectures. Journal of Systems and Software 59(3), 343–354.

- Coulette, B., T. D. Thu, X. Crégut, et D. T. B. Thuy (2000). Rhodes, a process component centered software engineering environment. In Proceedings of the International Conference on Enterprise Information Systems ICEIS'00, Stafford, UK, 2000, pp. 253–260.

- Crégut, X. et B. Coulette (1997). Pbool : an object-oriented language for definition and reuse of enactable processes. Software - Concepts and Tools 18(2), 47–62.

- Dami, S., J. Estublier, et M. Amiour (1997). APEL : a graphical yet executable formalism for process modeling. Automated Software Engineering Journal 5, 61–96.

- Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. Ph. D. thesis, University of California, Irvine, USA.

- Fuggetta, A. (2000). Software process : a roadmap. In Proceedings of the Conference on The Future of Software Engineering, ICSE'2000, Limerick, Ireland, 2000 Gallardo, L. (2000). Une approche à base de composants pour la modélisation des procédés logiciels. Ph. D. Thesis.

- Garlan, D. (2000). Software architecture : a roadmap. In Proceedings of the Conference on The Future of Software Engineering, ICSE '00, New York, NY, USA, pp. 91–101. ACM.

- David Garlan, D. (1995). Research directions in software architecture. ACM Computing Surveys, 27(2):257{261, June 1995.

- Gary, K., T. E. Lindquist, H. Koehnemann, et J.-C. Derniame (1998). Component-based software process support. In ASE, pp. 196–199.

- Hitomi, A. S., G. A. Bolcer, et R. N. Taylor (1997). Endeavors : a process system infrastructure. Proceedings of the 19th international conference on Software engineering, Bostgon, May 17-23, 1997, pp. 598-599.

- Hug, C. (2009). Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information. Ph. D. thesis, Université Joseph Fourier- Grenoble I, France.

- Kaba A.B., Tankoano J., D. J. (1994). Une approche incrémentale d'évolution des modèles de procédés de développement de logiciels. In Proceedings of the second African Conference on research in computer science, CARI'94, pp. 351–365.

- Leymonerie, F. (2004). ASL : un langage et des outils pour les styles architecturaux : contribution à la description d'architectures dynamiques. Thèse de doctorat, Université de Savoie, Décembre 2004..

- Medvidovic, N., P. Grünbacher, A. Egyed, et B. W. Boehm (2003). Bridging models across the software lifecycle. J. Syst. Softw. 68, 199–215.

- Medvidovic, N. et R. N. Taylor (1997). A framework for classifying and comparing architecture description languages. In ESEC / SIGSOFT FSE, pp. 60–76.

- Mehta, N. R., N. Medvidovic, et S. Phadke (2000). Towards a taxonomy of software connectors.

- In ICSE '00 : Proceedings of the 22nd international conference on Software engineering, pp. 178–187, Limerick, Ireland, 2000.

- Mishali, O. et S. Katz (2006). Using aspects to support the software process : Xp over eclipse. In Proceedings of the 5th international conference on Aspect-oriented software development,AOSD '06, New York, NY, USA, pp. 169–179. ACM.

- Object Management Group, OMG. (2008). Software Systems Process Engineering Metamodel, v2.0.

- Osterweil, L. (1987). Software processes are software too. In Proceedings of the 9th international conference on Software Engineering, ICSE '87, IEEE CS Press, 1987, pp. 2–13.

- Oussalah, M. (2005). Ingénierie des composants logiciels, Principes et fondements. Vuibert. Perry, D. E. et A. L.Wolf (1992). Foundations for the study of software architecture. SIGSOFT Soft. Eng. Notes 17(4), 40–52.

- Sanlaville, S. D. (2005). Environnement de procédé extensible pour l'orchestration Application aux services web. Ph. D. thesis.

- Sommerville, I. et T. Rodden (1996). Human, social and organisational influences on the software process. In: A.Fuggeta and A.Wolf (eds.): Trends in Software: Software Process. John Wiley and Sons, Chapt. 4.

- Starke, G. (1994). Why is process modelling so difficult ? In EWSPT '94 : Proceedings of the Third European Workshop on Software Process Technology, London, UK, pp. 163–166. Springer-Verlag.

- Szyperski, C. (2002). Component Software : Beyond Object-Oriented Programming (2nd ed.). Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc.

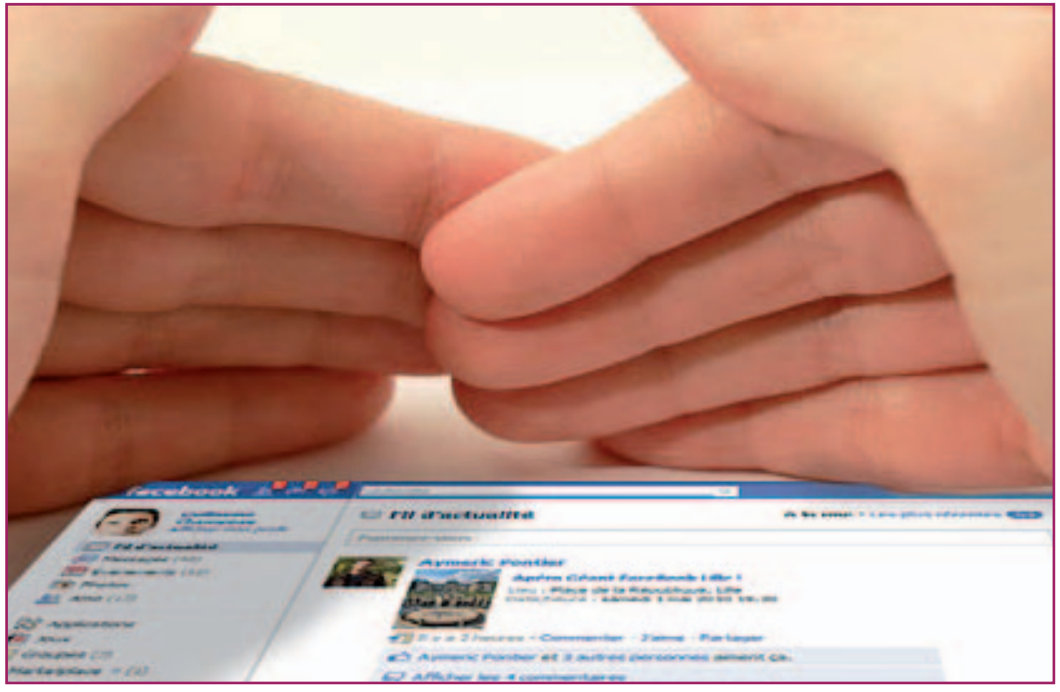
- Turk, D., R. France, et B. Rumpe (2000). Limitations of agile software processes. In Third International Conference On Extreme Programming And Flexible Processes In Software Engineering (XP2002), pp. 43–46. Springer-Verlag.

- Wang, A. I. (2002). A process centred environment for cooperative software engineering. In Proceedings of the 14th international conference on Software Engineering and knowledge Engineering, SEKE '02, New York, NY, USA, pp. 469–472. ACM.

- Zamli, K. Z. (2004). A survey and analysis of process modeling languages. Malaysian Journal of Computer Science 17(2), 68–89.

- Zamli, K. Z. et PA. Lee (2001). Taxonomy of process modeling languages. In Proc. of the ACS/IEEE Inter. Conf. on Computer Systems and Applications (AICCSA›01) Beirut, Lebanon, June 2001.


LES CONSEILS
DE DZ-CERT



Protéger sa vie privée sur Internet

Avec l'émergence des blogs et des réseaux sociaux, le nombre d'informations personnelles accessibles en ligne augmente sans cesse. Un nouveau concept est né par analogie à notre environnement naturel, il s'agit de l'intimité numérique et le droit de préserver sa vie privée et ses données personnelles. Dans cette rubrique, nous vous proposons un ensemble de bonnes pratiques pour protéger vos données personnelles sur Internet.

Qu'est ce qu'une donnée à caractère personnel ?

Il s'agit principalement des informations qui permettent d'identifier soit directement, soit indirectement par recoupement d'informations, une personne, telles que :

- nom, prénom,
- photo,
- date de naissance,
- statut matrimonial,
- adresse postale, email, adresse IP d'ordinateur
- n° de sécurité sociale,
- n° de téléphone,
- n° de carte bancaire,
- plaque d'immatriculation du véhicule,
- élément d'identification biométrique,
- les données de géolocalisation
- etc.

Les Conseils de
DZ – CERT

L'IDENTITE NUMERIQUE ?

« L'identité numérique d'un individu est composée de données techniques (adresse IP, cookies…), de données personnelles institutionnelles (nom prénom, adresse, n° de tel, certificats…) et informelles (commentaires, notes, billets, photos…). Toutes ces bribes d'information composent une identité numérique plus globale qui caractérise un individu, sa personnalité, son entourage et ses habitudes. Ces petits bouts d'identité fonctionnent comme des gènes : ils composent l'ADN numérique d'un individu ».

Attention sur Internet : tout se garde, rien ne se perd !

Le Web a une mémoire : toute contribution de votre part sur un site ou un forum par exemple peut demeurer en ligne pendant des années tant que ce même site est en ligne. Il est également possible de retrouver des archives d'anciennes versions de sites.

Exemple : le site **WayBackMachine (Internet Archive)** conserve des versions antérieures de pages de sites et blogs depuis 1996.

- Réfléchir avant de publier ou de poster des informations, avis ou photos :

Avant de poster un message ou de diffuser une vidéo ou une photo, sachez que tout ce que vous diffuserez volontairement ayant un caractère privé pourra être visualisé par des milliers de personnes (inconnus) avec le risque que ces contenus soient détournés.

- Protéger vos mots de passe : choisissez des mots de passe compliqués et ne les communiquer à personne.
- Donner le minimum d'informations personnelles sur Internet.
- Vérifier vos traces sur Internet en utilisant un moteur de recherche pour découvrir quelles informations vous concernant circulent sur Internet.
- Prendre le temps de lire les Conditions Générales d'Utilisation avant de s'inscrire sur les réseaux sociaux.
- Sécuriser son compte sur les réseaux sociaux en paramétrant correctement son profil.
- Utiliser un pseudonyme connu seulement de vos proches.
- Prenez garde aux sites qui offrent prix et récompenses en échange de votre contact ou de toute autre information.
- Ne répondez jamais, et sous aucun prétexte, aux spammeurs.

Bonnes pratiques par rapport à l'usage des smartphones pour protéger votre vie privée :

• ne pas enregistrer dans le smartphone des informations confidentielles telles que des codes secrets (ex : accès à la banque en ligne), des codes d'accès (travail, ordinateur portable) afin de limiter les risques en cas de vol, piratage, ou usurpation d'identité ;

• mettre en place un délai de verrouillage automatique du téléphone en veille. En effet, en plus du code PIN, ce dispositif permet de rendre inactif (verrouiller) le téléphone au bout d'un certain temps, ce qui empêche la consultation des informations contenues dans le téléphone en cas de perte ou de vol ;

• activer si possible le chiffrement des sauvegardes du téléphone en utilisant les réglages de la plate-forme avec laquelle le téléphone se connecte. Cette manipulation garantira que personne ne sera en mesure d'utiliser les données figurant dans le smartphone ;

• installer un antivirus quand cela est possible ;

• ne pas télécharger d'applications de sources inconnues en privilégiant les plates-formes officielles ;

• vérifier à quelles données contenues dans le smartphone l'application installée va avoir accès ;

• lire les conditions d'utilisation d'un service avant de l'installer, et ne pas hésiter à consulter l'avis des autres utilisateurs ;

• régler les paramètres au sein du téléphone ou dans les applications de géolocalisation afin de toujours contrôler quand et par qui l'appareil peut être géolocalisé ;

• désactiver le GPS ou le WIFI après utilisation de l'application de géolocalisation;

Zoom Sur Un Proje

Abdelraouf Ouadjaout
Attaché de Recherche

Division Théorie et Ingénierie
des Systèmes Informatiques

Vérification des
Programmes Embarqués

Introduction

Chaque science a connu durant son histoire certains « grands challenges » qui ont donné vie à des siècles de recherches et de réflexions. La plupart de ces défis scientifiques prestigieux n'ont pas été résolus par des individus, mais leur solution était en réalité le fruit de la collaboration de plusieurs générations de chercheurs. En Mathématique par exemple, le fameux théorème de Fermat énoncé en 1637 a fait l'objet d'innombrables recherches pendant près de 350 ans. Il n'a été démontré qu'en 1995 par Andrew Wiles qui a su profiter des résultats partiels de ses prédécesseurs.

La vérification des programmes est l'un des plus anciens « grands challenges » de l'histoire de l'informatique, consistant à : « concevoir des outils automatiques employant des raisonnements mathématiques et logiques rigoureux afin de vérifier la consistance des programmes avec leurs spécifications internes et externes ». Cette question fondamentale date en réalité des premiers jours de l'informatique avec les travaux précurseurs de John von Neumann et Alan Turing durant les années 40. Depuis cette date et jusqu›aujourd›hui, cette problématique reste non résolue et ne cesse d›évoluer suite au développement remarquable des systèmes informatiques qui sont de plus en plus complexes.

Néanmoins, les acteurs industriels et politiques ne se sont penchés sur cette question qu'à la suite de quelques épisodes « noirs » de l'histoire de l'informatique causés par des erreurs de programmation dans des logiciels critiques. Par exemple, le vol inaugural de la fusée Ariane 5 en 1996 s'est soldé par un échec à cause d'une erreur de conversion d'un flottant vers un entier. La fusée s'est brisée et a explosé en vol 40 secondes après son décollage, engendrant une perte de plus de 500 millions de dollars. Il existe malheureusement beaucoup d'autres exemples similaires qui ont causé des pertes humaines et matérielles considérables. Les gouvernements et les acteurs économiques ont décidé par la suite d'investir dans ce domaine de recherche afin de développer des méthodes permettant d'assurer le bon fonctionnement des programmes informatiques critiques.

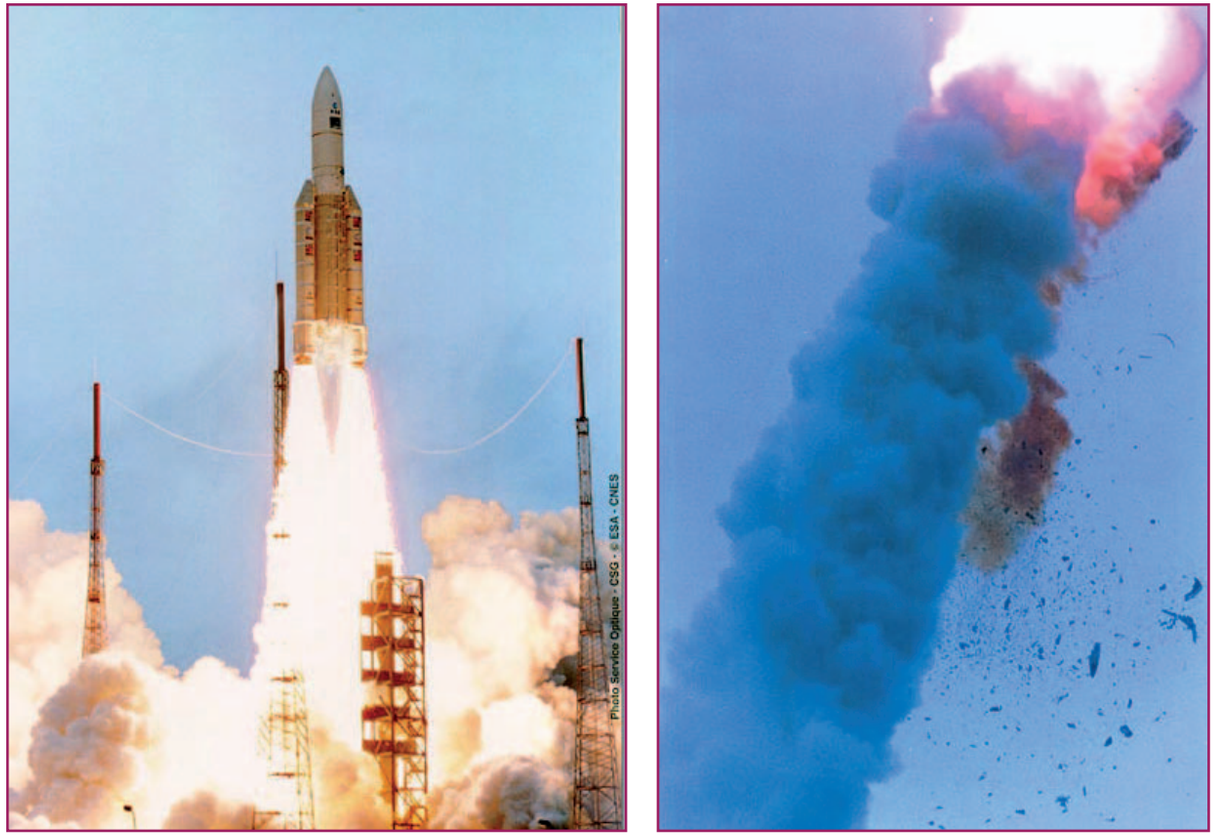


Figure 1 : Echec du vol inaugural d'Ariane 5 à cause d'un programme défaillant

Méthodes

La difficulté majeure de cet axe de recherche se résume dans un seul point central : la plupart des questions sur les comportements d'un programme ne sont pas décidables. En d'autres termes, il est impossible de tester – algorithmiquement et dans un temps fini - si tous les chemins d'exécution d'un programme vérifient une certaine propriété de sûreté. Cet obstacle inné à la nature des programmes informatiques est lié au fameux problème de l'arrêt qui a été prouvé comme étant non décidable par Alan Turing en 1936.

Les premières ébauches de solutions ont commencé avec les travaux de Robert Floyd, Peter Naur et C. A. R. Hoare durant les années 60. Ces méthodes sont basées sur des preuves déductives et nécessitent une intervention humaine. Par la suite, plusieurs techniques ont repris les fondements de ces idées pour automatiser le processus de vérification. Deux approches ont émergé :

- La première famille de solutions consiste à énumérer un sous-ensemble fini des chemins d'exécutions. Ce sont des méthodes dites de tests et se basent généralement sur des techniques issues du Model Checking. Ces méthodes sont utilisées pour prouver la présence d'erreurs et non pas leur absence. En effet, aucune conclusion ne peut être faite lorsque l'outil ne détecte pas d'erreurs, car l'espace d'états a été partiellement parcouru.
- La seconde approche consiste à omettre quelques détails sur les chemins d'exécution en opérant sur une sémantique approximative et grossière du programme. Les propriétés omises sont celles qui n'interfèrent pas avec la propriété de sûreté recherchée. En réduisant la précision de l'analyse, ces méthodes permettent de considérer tous les chemins possibles d'exécution dans un temps fini. Cette approche a été formalisée par Patrick Cousot et Radhia Cousot à travers la théorie de l'Interprétation Abstraite. Contrairement à la première approche, ces approximations sont utilisées pour prouver l'absence d'erreurs et non pas leur présence.

Notre Projet

Notre projet s'inscrit dans la thématique de vérification des programmes et s'intéresse plus particulièrement aux systèmes embarqués. Ce sont des systèmes autonomes qui interagissent avec un environnement externe pour effectuer une tâche précise. Ces systèmes sont destinés à opérer sans interruption pendant de longues durées sans intervention humaine. Ils doivent donc s'auto-organiser et être capables de gérer les éventuelles exceptions durant leur fonctionnement.

Il existe une très large panoplie d'applications des systèmes embarqués dans divers secteurs : l'environnement, l'industrie, la médecine, etc. Par exemple, notre équipe travaille actuellement sur un projet PNR consistant à employer la technologie des réseaux de capteurs sans

fil – qui est une sous-famille des systèmes embarqués – pour automatiser l'irrigation agricole et rationaliser les quantités d'eau utilises.

Figure 2 : Les réseaux de capteurs sans fil et quelques exemples d'applications

Ce système est constitué d'un ensemble de capteurs intelligents (appelés motes) déployés dans un champ agricole. Ces motes récoltent périodiquement le taux d'humidité du sol ainsi que quelques paramètres météorologiques tels que la température et la pression atmosphérique. La collecte de ces informations se fait d'une manière distribuée via des communications sans fil à courte portée. Les données sont par la suite consolidées et les zones ayant besoin d'eau sont irriguées à travers un ensemble d'électrovannes.

Les challenges rencontrés afin de vérifier ce genre de systèmes embarqués peuvent être résumés en ce qui suit :

- Ces logiciels embarqués sont caractérisés par un fort couplage avec la plate-forme matérielle. En d'autres termes, les interactions avec les couches matérielles se font à un bas niveau avec un accès direct aux registres et aux mémoires d'entrées/sorties. Afin de vérifier ces programmes, il est donc nécessaire de modéliser fidèlement le comportement des composants matériels. On est donc confronté à une vérification dite de systèmes considérant l'interaction du programme avec le matériel, contrairement à la vérification traditionnelle des programmes où on s'intéresse seulement à la manipulation de la mémoire.

- La nature distribuée des communications complique considérablement la tâche de vérification. En effet, un réseau de capteurs peut être considéré comme étant un ensemble de processus séquentiels interagissant via des messages. Afin de vérifier une propriété de sûreté sur le comportement global du système, on est vite confronté à une explosion combinatoire de l'espace d'états engendrée par le produit des espaces d'état de chaque processus.

L'objectif de notre projet est de contribuer à répondre à la question suivante : comment s'assurer qu'un programme embarqué interagit bien avec son matériel et que la collaboration en réseau de ces programmes s'effectue correctement ? Cette problématique est d'autant plus importante que l'Internet des Objets est considéré comme un futur proche. En connectant les objets de notre vie quotidienne au réseau internet, on pourrait voir une émergence des applications distribuées embarquées. Ces applications pouvant servir dans des domaines critiques, il est donc impératif de certifier leur bon fonctionnement.

The 1st International Conference on Information and Communication Technologies for Disaster Management, ICT-DM'2014 - Algiers, Algeria

March 24-25 2014

CALL FOR PAPER

Disasters are events that cause dramatic losses of life and property and disrupt the normal functioning of the economy and society on a large scale. Disaster management is multifaceted process to reduce the impact of disasters and provide assistance to affected populations. It involves information- and communication-intensive activities before, during and after disaster strikes. The revolutionary potential of Information and Communication Technologies (ICTs) lies in their ability to instantly and continuously facilitate rapid communication and flow of information, capital, ideas, people and products. Because of this potential, ICTs have shown usefulness for addressing various challenges including those related to disaster management. This makes everyone agree on their potential to revolutionize the disaster management area too.

ICT-DM'2014 aims to bring together academics and practitioners who are involved in emergency services, ad hoc planning, disaster recovery, etc., to learn about the latest research developments, share experiences and information about this area and develop recommendations.

There will also be invited presentations by experts from academia, industry, and government as well as special sessions dedicated to case studies, demonstrations and experiences based on pragmatic approaches. This way, the conference will provide a forum for supporting and encouraging both academic researchers, as well as practitioners involved in practice-focused research.

Authors are invited to submit manuscripts that present original unpublished research on using ICTs for detection, prevention, preparation, response and recovery of disasters. There will also be invited presentations by experts from academia, industry, and government as well as special sessions dedicated to case studies, demonstrations and experiences based on pragmatic approaches. This way, the conference will provide a forum for supporting and encouraging both academic researchers, as well as practitioners involved in applied-focused research.

TOPICS

The conference topics include (but are not limited):

1. Communication systems for disaster management
2. Coordination and collaboration technologies and systems for disaster management
3. Data/information management and analysis for disaster management
4. Geo-Information technologies for disaster management
5. Preparation and mitigation for managing disasters
6. Practices for risk reduction and rapid response in developing countries

■ DEMOS

ICT-DM'2014 will include demonstration sessions that shall provide a forum to present and discuss: demonstrations of new products, applications and techniques, practical implementations; industrial and commercial developments, research testbeds and implementations meeting the topics of the conference. A 2-page extended abstract describing the main contributions of the demo and the merits of the proposed ideas must be submitted.

■ *IMPORTANT DATES*

Paper Submission : September 23, 2013
Demo Submission : December 23, 2013
Author Notification : December 01, 2013
Demo Notification : January 20, 2014
Camera Ready : December 16, 2013
Conference Dates : March 24-25, 2014

Website: http://www.ict-dm.org
Contact: ictdm2014@gmail.com
Facebook: https://www.facebook.com/IctDm2014
Twitter: https://twitter.com/ictdm
Linkedin : http://www.linkedin.com/groups/ICTDM-2014-International-Conference-on-4970712?home=&gid=4970712

FORMATION

Le CERIST a abrité un atelier de formation sur le protocole Internet IPv6 organisé par l'Agence Universitaire de la Francophonie durant la période du 18 au 22 novembre 2012. Huit chercheurs et ingénieurs du centre ont bénéficié de cette formation ainsi que ceux d'autres institutions nationales et internationales telles que l'Université de Meknès, l'ISET Charguia, le CNF de Dakar, l'Université Saint Joseph Beyrouth, le CNM Tripoli, l'IFAO Caire, Algérie Télécom, l'USTHB et le Ministère des Postes et de la Technologie de l'Information et de la Communication.

Soutenance de thèse de Doctorat

Mme Bessai Fatma zohra, chargée de recherche et chef de l'équipe « Bases de données», a soutenu sa thèse de doctorat intitulé : « Modèle possibiliste pour la recherche d'information agrégée dans des corpus de documents semi structurés » le mercredi 12 décembre 2012 avec mention très honorable.

Cycle de conférences / **www.cerist.dz/conf**

- Dr Cheriet Farida Polytechnique Montréal CANADA : « La réalité virtuelle » 08 octobre 2012
- Dr Bagula (Lecturer in the Department of Computer Science of the University of Cape Town, South Africa) : 28 novembre 2012 « Ubiquitous Sensor Networking For Development (USN4D) »
- Dr. Anne V.D.M. Kayem Senior lecturer at the University of Cape Town, South Africa. 28 novembre 2012 "On Why Self-Protecting Access Control Is Good but Not Free"
- Pr. BENSALEM Saddek (Professor at the University of Joseph Fourier in Grenoble France): 09 décembre "Rigorous Component-based System Design Using the BIP Framework"

RAPPORTS DE RECHERCHE INTERNES

(http : // www.cerist.dz/publications)

Nekri Mounira, Khelladi Abdelkader, Proposition of structural approach for partitioning the web into communities. Alger: CERIST: 2012. ISRN CERIST-DRDSI/RR--12-000000039—dz

http://www.cerist.dz/publication/index.php?option=com_content&task=view&id=658&Itemid=53

Kouici Salima, A Clustering Application for the Web Usage Mining. Alger: CERIST: 2012. ISRN CERIST-DRDSI/RR--12-000000038—dz

http://www.cerist.dz/publication/index.php?option=com_content&task=view&id=659&Itemid=53

Chalabi Lydia, Characteristics and use of grey literature in scientific journals articles of Algerian researchers: Case study of University of Science and Technology Houari Boumediene (Physic, Chemistry and computer sciences). Alger: CERIST: 2012. ISRN CERIST-DRDSI/RR--12-000000032—dz

http://www.cerist.dz/publication/index.php?option=com_content&task=view&id=660&Itemid=53

Harik Hakim ; Belbachir Hacene, A new graph coloring and Fibonacchi number. Alger: CERIST: 2012. ISRN CERIST-DRDSI/RR--12-000000033—dz

http://www.cerist.dz/publication/index.php?option=com_content&task=view&id=661&Itemid=53

Setitra Insaf, Object classification in videos An overview. Alger: CERIST: 2012. ISRN CERIST-DSISM/RR--12-000000037-1—dz

http://www.cerist.dz/publication/index.php?option=com_content&task=view&id=662&Itemid=53

CERIST

Bases de données documentaires

Accessibles sur : www.cerist.dz

SYSTÈME NATIONAL DE DOCUMENTATION EN LIGNE

Le CERIST permet l'accès à une documentation électronique nationale et internationale couvrant tous les domaines scientifiques et techniques grâce au Système National de la Documentation en Ligne (SNDL).
Ce système concerne les chercheurs, les enseignants chercheurs et les étudiants.

De plus amples informations sont disponibles sur le site
www.sndl.cerist.dz

Directeur de publication

Pr. BADACHE Nadjib

Dossier : LES ARCHITECTURES DE PROCÉDÉS LOGICIELS : ETAT DE L'ART

Réalisé Par : **Pr. Mohamed Ahmed-Nacer**

Université des Sciences et de la Technologie Houari Boumediène

Dr. Fadila Aoussat

Université des Sciences et de la Technologie Houari Boumediène

Pr. Mourad Oussalah

Université de Nantes

Rubrique : Les Conseils de DZ - CERT

L'ÉQUIPE DZ-CERT

Rubrique : Zoom sur un Projet

Abdelraouf Ouadjaout

Comité de communication et de rédaction

BEBBOUCHI Dalila

BENNADJI Khedidja

DJETTEN Fatiha

Photographies

ALIMIHOUB Dahmane

Réalisation graphique

BOUKEZOULA Mohamed Amine

BENAKILA Nawel

Publié par le CERIST

5, rue des 3 Frères Aissou. Ben Aknoun. BP 143, 16030 - Alger

Tél : +213 (21) 91 62 05 – 08 / Fax : +213 (21) 91 21 26

E - mail : vrr@mail.cerist.dz

www.cerist.dz

Impression

ANEP

ISSN : 2170-0656 / **DÉPÔT LÉGAL :** 2690-201

Le Bulletin CERISTNEWS

CENTRE DE RECHERCHE SUR L'INFORMATION SCIENTIFIQUE ET TECHNIQUE - CERIST
5, Rue des Trois Frères Aissou, Ben - Aknoun - BP 143. 16030 - Alger
Tél : +213 (21) 91 62 05 - 08 / Fax : +213 (21) 91 21 26

www.cerist.dz