RECHERCHE • TECHNOLOGIE • APPLICATIONS

Informatique et Systèmes d'Information

Architectures logicielles

principes, techniques et outils

sous la direction de

Mourad Chabane Oussalah

Hermes

Lavoisier

Architectures logicielles

© 2014, Lavoisier, Paris
www.editions.lavoisier.fr

ISBN 978-2-7462-4517-4
ISSN 2111-0360

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, d'une part, que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective" et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite" (article L. 122-4). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Tous les noms de sociétés ou de produits cités dans cet ouvrage sont utilisés à des fins d'identification et sont des marques de leurs détenteurs respectifs.

Architectures logicielles

principes, techniques et outils

sous la direction de
Mourad Chabane Oussalah

hermes Science publications

Lavoisier

Série Informatique et Systèmes d'Information
sous la direction de Jean-Charles Pomerol

Les cartes à puce, Samia Bouzefrane et Pierre Paradinas, 2013

Mise en œuvre de la méthode B, Jean-Louis Boulanger, 2013

Utilisations industrielles des techniques formelles : interprétation abstraite, Jean-Louis Boulanger, 2011

Techniques industrielles de modélisation formelle pour le transport, Jean-Louis Boulanger, 2011

Interaction homme-machine dans les transports – personnalisation, assistance et informations du voyageur, Christophe Kolski, 2010

Liste des auteurs

Mathieu ACHER
INRIA/IRISA
Rennes

Mohamed AHMED-NACER
USTHB
Alger
Algérie

Abdelkrim AMIRAT
Centre Universitaire de Souk-Ahras
Algérie

Fadila AOUSSAT
LINA
Université de Nantes

Hugo ARBOLEDA
Université Icesi
Colombie

Jean-Paul ARCANGELI
IRIT
Université Paul Sabatier
Toulouse

Kamel BARKAOUI
CEDRIC
Cnam
Paris

Rubby CASALLAS
Université de Los Andes
Colombie

Jaime CHAVARRIAGA
Université de Los Andes
Colombie

Philippe COLLET
I3S
Nice Sophia Antipolis

Khalil DRIRA
LAAS
Université de Toulouse

Maryam ESLAMICHALANDAR
CEDRIC
Cnam
Paris

Anthony HOCK-KOON
Amadeus
Nice Sophia Antipolis

Ahmad KHEIR
LaMA Laboratory
Université Libanaise
Tripoli
Liban

Philippe LAHIRE
I3S
Nice Sophia Antipolis

Nicole LEVY
CEDRIC
Cnam
Paris

Francisca LOSAVIO
MoST
Université Centrale du Venezuela
Caracas
Venezuela

José Carlos MALDONADO
ICMC
Université de São Paulo
Brésil

Frédéric MIGEON
IRIT
Université Paul Sabatier
Toulouse

Elisa Yumi NAKAGAWA
ICMC
Université de São Paulo
Brésil

Hala NAJA
LaMA Laboratory
Université Libanaise
Tripoli
Liban

Victor NOËL
IRIT
Université Paul Sabatier
Toulouse

Flavio OQUENDO
IRISA
Rennes

Mourad Chabane OUSSALAH
LINA
Université de Nantes

Yann POLLET
CEDRIC
Cnam
Paris

Jean-Claude ROYER
LINA
Ecole des Mines de Nantes

Maria-Teresa SEGARRA
Télécom Bretagne
Brest

Adel SMEDA
Université de Tripoli
Tripoli
Lybie

Chouki TIBERMACINE
LIRMM
Montpellier

Mohamed ZOUARI
LAAS
Université de Toulouse

Table des matières

Avant-propos 15
Mourad Chabane OUSSALAH

Chapitre 1. Paradigmes objet, composant, agent et service dans les architectures logicielles 19
Abdelkrim AMIRAT, Anthony HOCK-KOON et Mourad Chabane OUSSALAH

1.1. Introduction. 19
1.2. Historique. 20
1.2.1. Paradigme objet 21
1.2.2. Paradigme composant 22
1.2.3. Paradigme agent. 22
1.2.4. Paradigme service. 23
1.3. Architecture logicielle 25
1.3.1. Architecture logicielle à base d'objets. 25
1.3.2. Architecture logicielle à base de composants. 27
1.3.3. Architecture logicielle à base d'agents 28
1.3.4. Architecture logicielle orientée services (AOS) 30
1.4. Les deux dimensions du cadre conceptuel de comparaison : quantitative et qualitative 32
1.4.1. Différences conceptuelles 33
1.4.2. Dimension quantitative. 42
1.4.3. Dimension qualitative 48
1.5. Approches d'intégration de paradigmes de développement 58
1.6. Synthèse et discussion 60
1.7. Conclusion 62
1.8. Bibliographie. 63

Chapitre 2. Architectures logicielles : contraintes d'architecture 69
Chouki TIBERMACINE

2.1. Introduction. 70
2.2. Etat de l'art 72
2.2.1. Expression de contraintes d'architecture en phase de conception 72
2.2.2. Expression de contraintes d'architecture en phase d'implémentation 80
2.3. Contraintes d'architecture sur des applications à objets. 87
2.3.1. Contraintes d'architecture en phase de conception 87
2.3.2. Contraintes d'architecture en phase d'implémentation 91
2.4. Contraintes d'architecture sur des applications à composants 97
2.4.1. Contraintes d'architecture en phase de conception 97
2.4.2. Contraintes d'architecture en phase d'implémentation 103
2.5. Contraintes d'architecture sur des applications à services 107
2.6. Conclusion 112
2.7. Bibliographie. 113

Chapitre 3. Architectures logicielles et variabilité multiple. 119
Mathieu ACHER, Philippe COLLET et Philippe LAHIRE

3.1. Introduction. 119
3.2. Variabilité : fondements et principes 123
3.2.1. Variabilité et lignes de produits 123
3.2.2. *Feature models* 125
3.3. Cadre d'études et travaux connexes. 127
3.3.1. De la multiplicité de la variabilité 127
3.3.2. Extraction et évolution de la variabilité architecturale. 128
3.4. Architecture de composants pour la vidéosurveillance 130
3.4.1. Etude de cas 130
3.4.2. Prise en compte de la variabilité multiple 130
3.4.3. Résultats 134
3.5. Architecture orientée services pour les *workflows* scientifiques 136
3.5.1. Etude de cas 136
3.5.2. Prise en compte de la variabilité multiple 137
3.5.3. Résultats 140
3.6. Rétro-ingénierie d'architecture à *plugins* 141
3.6.1. Etude de cas 141
3.6.2. Prise en compte de la variabilité multiple 143
3.6.3. Résultats 146

3.7. Bilan 147
3.7.1. De la nécessité d'un outillage 147
3.7.2. Synthèse des études de cas. 149
3.8. Conclusion 150
3.9. Bibliographie 151

Chapitre 4. Les architectures de référence. 159
Elisa Yumi NAKAGAWA, Flavio OQUENDO et José Carlos MALDONADO

4.1. Introduction. 159
4.2. Définition de l'architecture de référence. 160
4.2.1. Architecture de référence et modèle de référence 161
4.2.2. Architecture de référence et architecture de ligne de produits 162
4.3. Un modèle d'architecture de référence. 163
4.4. Ingénierie d'architecture de référence 167
4.4.1. Investigation des sources d'informations 168
4.4.2. Etablissement d'exigences architecturales 170
4.4.3. Conception d'architectures de référence 171
4.4.4. Evaluation des architectures de référence. 173
4.5. Usages des architectures de référence 175
4.6. Exemples d'architectures de référence. 179
4.7. Perspectives d'avenir pour les architectures de référence. 180
4.8. Conclusion 183
4.9. Bibliographie. 183

Chapitre 5. Architectures logicielles pour les lignes de produits 189
Hugo ARBOLEDA, Rubby CASALLAS, Jaime CHAVARRIAGA
et Jean-Claude ROYER

5.1. Introduction aux lignes de produits logiciels 189
5.1.1. Trois styles de développements. 192
5.1.2. La gestion de la variabilité. 192
5.1.3. La notion d'architecture dans les lignes de produits. 194
5.2. L'exemple du dépôt de musique 196
5.2.1. Le domaine. 196
5.2.2. La ligne de produits SongStock. 197
5.2.3. Les besoins fonctionnels. 197
5.2.4. Les principaux autres besoins. 198
5.3. Ingénierie du domaine 199
5.3.1. L'analyse du domaine 199
5.3.2. Cas d'utilisation avec variabilité intégrée. 200

5.3.3. Le modèle des caractéristiques 201
5.3.4. La conception du domaine. 202
5.3.5. La conception de l'architecture de la ligne 203
5.4. Ingénierie des produits 207
5.4.1. La configuration d'un produit. 208
5.4.2. La dérivation du produit 208
5.5. Processus de conception de l'architecture de référence 210
5.6. Lectures complémentaires 213
5.6.1. PLA et architecture de référence 214
5.6.2. Les livres des précurseurs 215
5.7. Conclusion 218
5.8. Bibliographie. 219

Chapitre 6. Architectures logicielles multihiérarchies/multivues. 227
Ahmad KHEIR, Hala NAJA et Mourad Chabane OUSSALAH

6.1. Introduction. 227
6.2. Approches de points de vue existantes. 228
6.2.1. Introduction 228
6.2.2. Les vues en spécification des besoins 229
6.2.3. Les vues en modélisation de systèmes. 229
6.2.4. Les vues en programmation 230
6.3. Les vues en architecture logicielle 231
6.3.1. Apports des vues en architecture logicielle 231
6.3.2. Le modèle « 4+1 » *View Model*. 232
6.3.3. Le standard ISO/IEC/IEEE 42010 233
6.3.4. L'approche *Views and beyond* 235
6.3.5. Synthèse 237
6.3.6. Limitations des approches actuelles en architecture logicielle. . . . 239
6.4. Définitions et concepts de base des architectures logicielles multihiérarchies/multivues 240
6.4.1. Définitions 240
6.4.2. Concepts et techniques de base 242
6.5. MoVAL : une architecture à base de modèles, vues et niveaux d'abstraction 251
6.5.1. Introduction 251
6.5.2. MoVAL. 251
6.5.3. Le métamodèle MoVAL. 257
6.5.4. Etude de cas 258
6.6. Conclusion 262
6.7. Bibliographie. 263

Chapitre 7. Architecture et qualité de systèmes logiciels 265
Nicole LEVY, Francisca LOSAVIO et Yann POLLET

7.1. Introduction. 265
7.2. La démarche qualité. 267
7.2.1. Le standard de qualité ISO 25010 267
7.2.2. Le référentiel de qualité 270
7.2.3. Modèle de qualité d'un système 271
7.2.4. Modèle de qualité fonctionnelle 271
7.2.5. Modèle de qualité d'une architecture 272
7.3. Démarche de développement d'une architecture de domaine 274
7.3.1. Principes généraux 274
7.3.2. Le modèle de qualité fonctionnelle. 277
7.3.3. Le modèle de qualité architecturale 277
7.3.4. Architecture de référence 277
7.3.5. Passage du niveau domaine fonctionnel au niveau système 279
7.4. Développement d'une architecture de référence
d'un domaine fonctionnel 280
7.4.1. Un exemple de domaine fonctionnel. 280
7.4.2. Le raffinement fonctionnel 280
7.4.3. Elaboration du modèle de qualité fonctionnelle 282
7.4.4. Définition de l'architecture préliminaire 283
7.4.5. Elaboration du modèle de qualité architecturale 284
7.4.6. Intégration de l'architecture de référence du domaine 285
7.5. Architectures de niveau système 288
7.5.1. Le raffinement fonctionnel 289
7.5.2. Le modèle de qualité fonctionnelle. 290
7.5.3. Architecture de base 290
7.5.4. Le modèle de qualité architecturale 291
7.5.5. Architecture des systèmes Dopamine et Samarkand 292
7.6. Travaux connexes 294
7.7. Conclusion 299
7.8. Bibliographie. 300

**Chapitre 8. Architectures logicielles et outils : gestion distribuée
et coordonnée de la reconfiguration dynamique**. 303
Mohamed ZOUARI, Maria-Teresa SEGARRA et Khalil DRIRA

8.1. Introduction. 304
8.2. Contexte. 305

8.3. Mécanismes de gestion de la reconfiguration dynamique d'applications distribuées 307
8.3.1. Gestion centralisée de la reconfiguration dynamique 307
8.3.2. Limites de la solution centralisée pour les systèmes distribués 310
8.3.3. Enjeux de la gestion distribuée de la reconfiguration 311
8.3.4. Mécanismes de coordination existants 312
8.4. Spécialisation des infrastructures de reconfiguration. 315
8.4.1. Spécialisation de comportement 315
8.4.2. Spécialisation de la répartition des mécanismes d'adaptation 316
8.5. Synthèse des limites et des défis de la reconfiguration dynamique des systèmes distribués 317
8.6. Approche pour la mise en place de mécanismes de gestion de reconfiguration 318
8.7. Modèle architectural pour la gestion distribuée de la reconfiguration dynamique 320
8.7.1. Types de composants pour la gestion de l'adaptation 320
8.7.2. Distribution de la gestion de la reconfiguration dynamique 322
8.7.3. Modèle architectural du gestionnaire d'adaptation. 324
8.7.4. Spécialisation des mécanismes de reconfiguration. 326
8.7.5. Coordination des processus de reconfiguration. 328
8.8. Conclusion 342
8.9. Bibliographie 343

Chapitre 9. La métamodélisation d'architectures logicielles. 347
Adel SMEDA et Mourad Chabane OUSSALAH

9.1. Introduction. 347
9.2. La métamodélisation : pourquoi ? 349
9.3. Métamodélisation d'architectures logicielles 350
9.4. MADL : un langage de description de méta-architecture 351
9.4.1. Les quatre niveaux de modélisation dans les architectures logicielles 351
9.4.2. MADL : un noyau réflexif dédié à la méta-méta-architecture 353
9.4.3. La structure de MADL 354
9.4.4. L'instanciation de MADL : l'exemple de l'ADL Acme 356
9.4.5. Comparaison de MADL et MDA/MOF 359
9.5. La projection (*mapping*) des ADL vers UML. 363
9.5.1. Pourquoi la projection des ADL vers UML. 363
9.5.2. La projection d'un ADL vers UML 365
9.6. Un exemple de projection : cas du langage Acme 376

9.7. Quelques remarques sur la projection des concepts d'ADL vers UML 377
9.7.1. UML 2.0 comme ADL 377
9.7.2. Les stratégies de la projection. 378
9.8. Conclusion 379
9.9. Bibliographie 379

Chapitre 10. Architectures logicielles et systèmes multi-agents 383
Jean-Paul ARCANGELI, Victor NOËL et Frédéric MIGEON

10.1. Introduction 383
10.2. Systèmes multi-agents et ingénierie logicielle à base d'agents 384
10.2.1. Agent 385
10.2.2. Système et interactions 386
10.2.3. Système multi-agents 387
10.2.4. Exemples de systèmes multi-agents 388
10.2.5. Ingénierie logicielle à base d'agents 390
10.3. Les systèmes multi-agents en tant que style d'architecture 394
10.3.1. Positionnement du style « système multi-agents » 394
10.3.2. Caractéristiques en matière d'abstraction 395
10.3.3. Caractéristiques en matière de (dé)composition 398
10.3.4. Lien avec les exigences. 400
10.3.5. Une famille de styles architecturaux 403
10.4. Le fossé architectural 404
10.4.1. Etat de la pratique 405
10.4.2. Analyse d'un point de vue architectural 406
10.4.3. Bilan 408
10.5. Comment combler le fossé architectural 409
10.5.1. Limites des solutions existantes. 409
10.5.2. Réalisation de l'architecture micro 409
10.6. Conclusion 412
10.7. Bibliographie 413

Chapitre 11. Architectures logicielles et procédés logiciels 417
Fadila AOUSSAT, Mourad Chabane OUSSALAH et Mohamed AHMED-NACER

11.1. Introduction 417
11.2. Les architectures de procédés logiciels 419
11.2.1. Définition des modèles de procédés logiciels 419
11.2.2. Modélisation de procédés logiciels à base d'architectures logicielles 420

11.3. Cadre de comparaison pour les solutions de réutilisation de modèles de PL à base d'AL 423
11.3.1. Les critères d'évaluation selon l'axe procédé logiciel 424
11.3.2. Les critères d'évaluation selon l'axe architecture logicielle 427
11.3.3. Les critères d'évaluation selon l'axe qualité 430
11.4. Evaluation des approches de modélisation et d'exécution de PL à base d'AL 433
11.4.1. Evaluation des approches de réutilisation de PL à base d'AL selon l'axe PL 433
11.4.2. Evaluation des approches de réutilisation de PL à base d'AL selon l'axe AL 435
11.4.3. Evaluation des approches de réutilisation de PL à base d'AL selon l'axe qualité 438
11.4.4. Bilan et discussions 440
11.5. Conclusion 442
11.6. Bibliographie 443

Chapitre 12. Architecture logicielle : adaptation de services dans le contexte de composition de services web 447
Kamel BARKAOUI et Maryam ESLAMICHALANDAR

12.1. Introduction 447
12.2. Composition de services web et de vérification 450
12.3. Incompatibilité des services web et adaptation 454
12.4. Approches d'adaptation 457
12.5. Conclusion 469
12.6. Bibliographie 470

Index 477

Avant-propos

Durant les deux dernières décennies, plusieurs langages de description d'architectures logicielles ont été proposés pour favoriser le développement d'applications centrées architectures. Ces langages fournissent, en général, des notations formelles ou semi-formelles pour décrire et analyser les systèmes logiciels. Ils sont généralement accompagnés d'outils destinés à analyser, simuler et parfois générer le code des systèmes modélisés. Les architectures logicielles ont apporté une réelle contribution dans le développement de systèmes complexes et distribués. Leurs principales caractéristiques résident d'une part dans leur pouvoir de gérer les abstractions et les niveaux d'expressivité d'un système, et d'autre part dans leur capacité à prendre en compte la modélisation de la structure et du comportement d'un système. Il est bien admis aujourd'hui qu'un des problèmes critiques dans la conception et l'élaboration de tout système logiciel complexe est son architecture, c'est-à-dire l'organisation des éléments architecturaux qui le constituent. Une bonne architecture peut aider à exhiber les propriétés cruciales d'un système (fiabilité, portabilité, interopérabilité, etc.). Au contraire, une mauvaise architecture peut avoir des conséquences désastreuses sur le système. Par ailleurs, une bonne architecture peut servir de « conscience » à un système au cours de son évolution. En effet, elle guide le processus d'évolution d'un système en indiquant par exemple quels aspects du système peuvent être modifiés sans compromettre son intégrité.

Aujourd'hui, les nouvelles applications d'ingénierie, en particulier celles dédiées à la conception et au développement de systèmes d'information distribués à base d'objets, de composants, de services, d'agents et de modèles ont mis en évidence le caractère évolutif des éléments architecturaux manipulés et des structures qui leur sont associées. La pérennité, l'adaptabilité et l'évolutivité de tels systèmes deviennent un enjeu économique très important. En effet, ces systèmes ont souvent nécessité plusieurs années (hommes/mois) de développement et de mise à jour qu'il devient impératif de leur assurer une durée de vie conséquente et surtout de répondre aux

évolutions de besoins des constructeurs d'architectures logicielles, des constructeurs d'applications et des utilisateurs finaux. Dans ce contexte, les communautés académiques et industrielles ont proposé la conception et le développement de nouvelles architectures logicielles ouvertes (c'est-à-dire capables de dialoguer et de s'interfacer avec d'autres architectures), adaptables (c'est-à-dire susceptibles d'être adaptées à des conditions opératoires et à des domaines d'application différents) et évolutives (c'est-à-dire capables d'être complétées/modifiées pour prendre en compte des services non prévus lors de leur conception).

En fait, les architectures logicielles constituent une bonne réponse aux besoins inhérents de systèmes complexes tels que :

– le besoin de pouvoir (ré)utiliser ces systèmes dans des environnements et des contextes différents. Il est ainsi nécessaire d'être capable de les remodéliser (évolution adaptative, évolutive et réingénierie) pour qu'ils puissent répondre d'une part, aux exigences et spécificités de leurs contextes d'utilisation, par exemple : infrastructures réparties, ressources limitées, points de vue différents sur la composition des différentes fonctionnalités, etc., et d'autre part, à de nouvelles exigences technologiques (composant, service, vue architecturale, etc.) ;

– le besoin de diminuer les coûts du développement et de maintenance de ces systèmes en les concevant par réutilisation et composition d'éléments architecturaux réutilisables (objets, composants, services et agents). Ces éléments architecturaux peuvent être identifiés et extraits à partir de systèmes existants pour être réutilisés dans de futurs développements ;

– le besoin de mettre à jour ces systèmes rapidement et avec le moindre coût. Dans ce cas, il est nécessaire d'automatiser les processus de transformation (structurelle et comportementale) et de favoriser la réutilisation de ces activités (réutilisation du processus d'évolution) ;

– le besoin de maîtriser leur complexité (comprendre ces systèmes, les faire évoluer, etc.) en les manipulant à un haut niveau d'abstraction.

Aujourd'hui, fort de ce constat, un certain nombre de propositions ont émergé sur les architectures à base de composants, les architectures à base de services (SOA), les architectures à base d'agents et les architectures à base de modèles. Dans tous les cas, les enjeux à termes sont l'amélioration de la qualité et de la productivité, et l'industrialisation de la production de logiciel.

Aussi, devant une telle diversité de propositions et de solutions, il était indispensable de faire le point sur la recherche et les applications concernant l'ingénierie des architectures logicielles.

C'est pourquoi, nous proposons un ouvrage, intitulé *Architectures logicielles*, qui s'intéresse aux paradigmes d'architectures logicielles à base d'objets, de composants, de services, d'agents et de modèles ; aux différentes techniques et méthodes architecturales, à l'analyse des qualités architecturales, aux modèles de représentation de patrons et styles architecturaux, leur formalisation, leur validation et finalement la démarche d'ingénierie dans laquelle ces éléments cohérents et autonomes peuvent être gérés.

Cet ouvrage s'adresse essentiellement aux DSI, chefs de projets, maîtres d'ouvrage, architectes, concepteurs, développeurs et utilisateurs d'architectures logicielles, aux enseignants d'écoles d'ingénieurs et d'universités, aux ingénieurs, aux étudiants de master et de doctorat et enfin aux élèves ingénieurs.

Mourad Chabane OUSSALAH