
Contents

Part I Introduction to Internetware

1	Internetware: A Shift of Software Paradigm	3
1.1	Introduction	3
1.2	The Internetware Paradigm	4
1.2.1	The Need of a Paradigm Shift	4
1.2.2	Methodological Reflections	5
1.2.3	Architecting Principles for Internetware	7
1.2.4	Our Roadmap and Explorations	8
1.3	Open Coordination Model	8
1.3.1	Coordination Model for Internetware	8
1.3.2	A Software Architecture Based Approach	10
1.4	Environment-Driven Model	11
1.4.1	Structure and Dynamics	11
1.4.2	Framework and Techniques	13
1.5	Dependability Assurance Framework with Trust-Management	14
1.5.1	A Trust-Based Approach	14
1.5.2	A Dependability Assurance Framework with Trust Management	15
1.6	Conclusion	15
	References	16
2	Technical Framework for Internetware: An Architecture Centric Approach	19
2.1	Introduction	19
2.2	Software Model of Internetware	21
2.2.1	Basic Component Model	22
2.2.2	Open Collaboration Model	22
2.2.3	Context Driven Model	23
2.2.4	Intelligent Trustworthy Model	23
2.3	Middleware for Internetware	23
2.3.1	The Implementation Model of Basic Internetware Entities	24
2.3.2	The Implementation Model of On-Demand Collaborations	24
2.3.3	The Autonomic Management of Middleware	25
2.3.4	Componentization of Middleware	25

2.4	Engineering Approach for Internetware	26
2.4.1	Software Architecture in the Whole Lifecycle	26
2.4.2	Development of Internetware Basic Entities and Their Collaborations	28
2.4.3	Domain Modeling for Internetware	28
2.5	Conclusion.	29
	References.	29

Part II Internetware Software Model

3	On Environment-Driven Software Model for Internetware	33
3.1	Introduction	33
3.2	An Approach to the Environment-Driven Model	35
3.2.1	Overall Interaction Patterns of Environment-Driven Applications	36
3.2.2	A Structuring Model for Environment-Driven Applications	37
3.2.3	Model and System Characteristics	40
3.3	Environment Modeling and Enabling Techniques	42
3.3.1	Context Processing Framework	43
3.3.2	Ontology-Based Modeling of Dynamic Context	44
3.3.3	Logic Distance-Based Context Retrieval.	46
3.3.4	Blurring Degree-Based Privacy Protection	48
3.3.5	Negotiation-Based Context Information Selection	50
3.4	Goal-Driven Adaptation and Rearon Ontologies	51
3.4.1	Rearon Ontologies.	52
3.4.2	Overall Interaction Patterns of Environment-Driven Applications	55
3.5	Prototypes and Experiments	56
3.5.1	Artemis-ARC	57
3.5.2	Artemis-MAC.	58
3.5.3	Artemis-FollowMeLite	60
3.5.4	Artemis-FollowMeAgent	61
3.5.5	Environment-Driven Applications	62
3.6	Related Work	64
3.7	Conclusion.	66
	References.	67
4	On Self-adaptation Model for Internetware	71
4.1	Introduction	71
4.2	Approach Overview	73
4.3	Key Techniques	74
4.3.1	Basic SA Model	74
4.3.2	SA Reflection.	76
4.3.3	SA Dynamism	76
4.3.4	SA Reasoning.	77

4.3.5	SA Trustworthiness	78
4.3.6	A Tool for Modeling SA in the Whole Lifecycle	80
4.4	Case Study.	81
4.4.1	Pattern Driven Self-adaptation.	81
4.4.2	Style Driven Self-adaptation	85
4.4.3	Performance Impact.	88
4.5	Related Work	89
4.6	Conclusion.	90
	References.	90
5	On Requirements Model Driven Adaption and Evolution of Internetwork.	93
5.1	Introduction	94
5.2	Matching and Composing Internetwork Components	95
5.2.1	A Component Meta Model.	95
5.2.2	Component Lifecycle.	97
5.2.3	Component Composition Rules.	98
5.3	Components Adaptation.	99
5.3.1	Adaptation Strategies	99
5.3.2	Adaptation Process	100
5.4	Internetwork Testbed and Case Studies	102
5.4.1	Internetwork Testbed	102
5.4.2	Model-Driven Development and Evaluation of Internetwork Applications	103
5.4.3	Enterprise Applications Example.	105
5.4.4	Experiment Scenarios on the Collaborations and Evolutions of Internetwork Components	105
5.5	Related Work	107
5.6	Conclusion and Future Work	108
	References.	109
	Part III Internetwork Operating Platform	
6	Runtime Recovery and Manipulation of Software Architecture of Component-Based Systems	115
6.1	Introduction	116
6.2	Approach Overview	118
6.2.1	Architectures in ABC.	118
6.2.2	Conceptual Framework	119
6.3	Characteristics of Software Architecture at Runtime	119
6.3.1	Software Architecture at Design Time	120
6.3.2	Software Architecture at Runtime	121
6.3.3	Description for Software Architecture Recovered at Runtime	123
6.4	Recovery of Software Architecture at Runtime	124
6.4.1	Overview of PKUAS.	124
6.4.2	Recovery of Basic Elements	126
6.4.3	Construction of Architecture Views	129
6.4.4	Enrichment of Semantics	130

6.5	Manipulation of Recovered Software Architecture at Runtime	131
6.5.1	Manipulation via Reflection	131
6.5.2	Programming Model	132
6.5.3	Graphical Tool	132
6.6	Performance Evaluation	134
6.7	Related Work	135
6.8	Conclusion and Future Work	136
	References	137
7	Supporting Runtime Software Architecture:	
	A Bidirectional-Transformation-Based Approach	139
7.1	Introduction	140
7.2	Runtime Software Architecture	141
7.2.1	An Illustrative Example	141
7.2.2	A Formal Description of Runtime Software Architecture	142
7.3	Maintaining Causal Connections by Architecture-System Synchronization	144
7.3.1	The Four Properties	144
7.3.2	The Challenges	144
7.4	Architecture-System Synchronization Based on Bi-transformation	145
7.4.1	Enabling Techniques	145
7.4.2	The Synchronization Algorithm	146
7.4.3	Assumptions	149
7.4.4	Discussion About the Algorithm and the Properties	149
7.5	Generating Synchronizers for Legacy Systems	150
7.5.1	Implementing the Generic Synchronization Engine	151
7.5.2	Generating Specific XMI Parsers and System Adapters	151
7.6	Case Studies	152
7.6.1	C2-JOnAS	152
7.6.2	Client/Server-JOnAS	154
7.6.3	Other Case Studies	155
7.6.4	Summary and Discussion	155
7.7	Related Work	158
7.8	Conclusion	159
	References	159
8	Low-Disruptive Dynamic Updating of Internetware Applications	163
8.1	Introduction	163
8.2	Dynamic Updating: Eager Versus Lazy	165
8.3	Javelus: Overview	166
8.4	Dynamic Patch Generation	167
8.4.1	Identifying Changed Classes	167
8.4.2	Default Transformation and Custom Transformers	168

8.5	Updating Code	170
8.5.1	Reaching a DSU Safe Point	170
8.5.2	Updating Classes.	170
8.6	Updating Objects	171
8.6.1	Affected Types	171
8.6.2	Optimizing Object Validity Checks	173
8.6.3	Mixed Object Model	174
8.6.4	Transforming Objects.	175
8.6.5	Continuous Updating.	176
8.7	Evaluation	176
8.7.1	Experiments with Micro Benchmarks.	177
8.7.2	Experiments with Tomcat and H2	179
8.8	Related Work	179
8.8.1	Dynamic Updating Systems for Java	179
8.8.2	Dynamic Updating Systems for Other Programming Languages	181
8.8.3	Dynamic Updating for Component-Based Systems	181
8.9	Conclusion.	181
	References.	182
9	Specification and Monitoring of Data-Centric Temporal Properties for Service-Based Internetwork Systems	185
9.1	Introduction	186
9.2	Motivation Example	187
9.2.1	Car Rental Application.	187
9.2.2	Supply Chain Application.	190
9.3	A Language for Specifying Data-Centric Properties.	191
9.3.1	The Syntax of Par-BCL	191
9.3.2	Par-BCL for Data-Centric Properties	194
9.4	Monitoring Model of Par-BCL	194
9.4.1	Filtering	194
9.4.2	Parameter Binding.	196
9.4.3	Verification	197
9.5	Efficient Monitoring Process	198
9.5.1	Parameter State Machine	198
9.5.2	Monitor Instance	199
9.5.3	Monitoring Algorithm	199
9.5.4	Management of Monitor Instances.	200
9.5.5	Architecture and Implementation	201
9.6	Experiments	202
9.6.1	Violation Detection	202
9.6.2	Performance Evaluation	202
9.6.3	Overhead Evaluation	204
9.7	Discussion	205

9.8	Related Work	205
9.8.1	Engineering Self-adaptive System	205
9.8.2	Runtime Monitoring of Web Services	206
9.8.3	Parametric Properties Monitoring for OO Systems	207
9.9	Conclusion.	207
	References.	208
10	Runtime Detection of the Concurrency Property in Asynchronous Pervasive Computing Environments	211
10.1	Introduction	212
10.2	Property Detection for Pervasive Context.	213
10.2.1	Asynchronous Message Passing and Logical Time	213
10.2.2	Consistent Global State (CGS) and the Lattice Structure Among CGSs	213
10.2.3	Specification of Contextual Properties	214
10.3	Concurrent Activity Detection in Asynchronous Pervasive Computing Environments	215
10.3.1	$Def(\phi)$ and Concurrent Contextual Activities	215
10.3.2	Design of the CADA Algorithm	218
10.3.3	Discussions	219
10.4	Performance Analysis	220
10.4.1	The Causes of Faults	221
10.4.2	Concurrency Among Sensed Activities.	221
10.4.3	Sensed and Physical Activities	223
10.4.4	Discussions	224
10.5	Experimental Evaluation	224
10.5.1	Implementation	224
10.5.2	Experiment Setup	225
10.5.3	Effects of Tuning the Update Interval	225
10.5.4	Effects of Tuning the Message Delay.	226
10.5.5	Effects of Tuning the Duration of Contextual Activities	227
10.5.6	Effects of Tuning the Number of Non-checker Processes.	227
10.5.7	Lessons Learned	227
10.6	Related Work	228
10.7	Conclusion.	228
	References.	229

Part IV Internetwork Engineering Approach

11	A Software Architecture Centric Engineering Approach for Internetwork	233
11.1	Introduction	233
11.2	Overview of ABC Methodology	236
11.3	Feature Oriented Requirement Modeling for Internetwork	239

11.4	Architecture Modeling of Self-adaptive Internetwork . . .	244
11.5	Reflective Middleware as Internetwork	
	Operating Platform	248
11.5.1	SA-Based Reflection	250
11.5.2	Autonomous Components.	252
11.6	Conclusion.	255
	References.	256
12	Toward Capability Specification of Internetwork Entity	
	Based on Environment Ontology	257
12.1	Introduction	257
12.2	Environment Ontology.	259
12.2.1	Upper Environment Ontology	259
12.2.2	Domain Environment Ontology.	261
12.2.3	Construction of Domain	
	Environment Ontology.	262
12.3	Environment Ontology Based Capability	
	Specification.	266
12.3.1	Elements of EnOnCaS	267
12.3.2	An Example Capability Profile	269
12.3.3	Capability Specification Generation	270
12.3.4	Internetwork Entity Discovery	271
12.4	Related Work	275
12.5	Conclusion.	277
	References.	277
13	Feature-Driven Requirement Dependency Analysis	
	and High-Level Software Design	279
13.1	Introduction	279
13.2	The Preliminaries	280
13.2.1	Definition of Features	281
13.2.2	Basic Attributes of Features	281
13.2.3	Operationalization of Features.	282
13.2.4	Responsibility Assignment	282
13.2.5	Resource Containers	283
13.3	Feature Dependencies	284
13.3.1	Refinement.	284
13.3.2	Constraint.	285
13.3.3	Influence	286
13.3.4	Interaction	287
13.4	Connections Between Dependencies	288
13.4.1	Refinement-Imposed Constraints	289
13.4.2	Constraint and Interaction.	289
13.4.3	Influence and Interaction	289
13.5	Verification of Constraints and Customization	291
13.5.1	Feature-Oriented Customization-Based	
	Requirement Reuse	291
13.5.2	Customization Through a Series	
	of Binding-Times	291

13.5.3	Three Verification Criteria	291
13.5.4	The Effectiveness of the Three Verification Criteria	292
13.6	Design of High-Level Software Architecture.	293
13.6.1	An Overview	293
13.6.2	Resource Container Identification	293
13.6.3	Component Seed Creation	294
13.6.4	Component Construction	294
13.6.5	Interaction Analysis.	294
13.7	Related Work	295
13.8	Conclusions and Future Work.	296
	Appendix 1	297
	Appendix 2	298
	Appendix 3	298
	References.	299
14	rcos: A Refinement Calculus of Internetwork Systems	301
14.1	Introduction	301
14.2	Semantic Basis	303
14.2.1	Programs as Designs	303
14.2.2	Refinement of Designs.	305
14.3	Syntax of rcos	305
14.3.1	Commands	306
14.3.2	Expressions	306
14.4	Semantics	306
14.4.1	Structure, Value and Object	307
14.4.2	Static Semantics	308
14.4.3	Dynamic Variables	311
14.4.4	Dynamic States.	311
14.4.5	Evaluation of Expressions	312
14.4.6	Semantics of Commands	312
14.4.7	Semantics of a Program	317
14.5	Object-Oriented Refinement	317
14.5.1	Object System Refinement	318
14.5.2	Structure Refinement	319
14.6	Refinement Rules	323
14.7	Conclusions	328
14.7.1	Related Work	328
14.7.2	Support UML-like Software Development	329
14.7.3	Limitation and Future Work	330
	References.	331

Part V Experiments and Practices of Internetwork

15	Refactoring Android Java Code for On-Demand Computation Offloading	337
15.1	Introduction	338
15.2	Design Pattern for Computation Offloading	339
15.2.1	The Source Structure and the Target Structure	339
15.2.2	Refactoring Steps Overview	340
15.2.3	An Illustrative Example	342

15.3	Implementation of DPartner	342
15.3.1	Detect Movable Classes	342
15.3.2	Generate Proxies	343
15.3.3	Transform App Classes	343
15.3.4	Cluster App Classes.	345
15.3.5	Determine the Computations to Be Offloaded	347
15.3.6	Offload Computations at Runtime	348
15.4	Evaluation	350
15.4.1	Experiment Setup	350
15.4.2	Performance of Refactoring	351
15.4.3	Comparison of App Performance.	351
15.4.4	Comparison of App Power Consumption	353
15.4.5	The Effect of On-Demand Offloading	354
15.4.6	Experiments on 3G Network.	354
15.5	Related Work	355
15.6	Conclusion and Future Work	357
	References.	357
16	Towards Architecture-Based Management of Platforms in Cloud	359
16.1	Introduction	360
16.2	Motivating Example	361
16.3	An Architecture-Based Model for Platform Management in Cloud	363
16.3.1	Approach Overview.	363
16.3.2	The Architecture-Based Meta-Model	363
16.3.3	Runtime Changes	365
16.4	Implementations of the Runtime Architecture-Based Model	365
16.5	Evaluation	366
16.5.1	Anti-pattern Detection	366
16.5.2	VM States Checking	368
16.6	Related Work	368
16.7	Conclusion and Future Work	370
	References.	371
17	Golden Age: On Multi-source Software Update Propagation in Pervasive Networking Environments.	373
17.1	Introduction	374
17.2	System Model	375
17.3	Update Propagation Strategies.	376
17.3.1	Random Spread.	377
17.3.2	Youngest Age.	377
17.3.3	Golden Age	377
17.4	Performance Analysis	378
17.4.1	Notations and Assumptions	378
17.4.2	Number of Updated Nodes in the System.	378
17.4.3	Principles for Choosing the Golden Age.	382

17.5	Numerical Results	383
17.5.1	Simulation Setup	383
17.5.2	The Number of Updated Nodes	383
17.5.3	Impact of Source Density	384
17.5.4	Impact of Buffer Size	384
17.5.5	Impact of Update Interval	385
17.5.6	Performance of Different Golden Age Parameter Choosing Rules	385
17.6	Related Work	385
17.7	Conclusion	386
	References	387
18	GreenDroid: Automated Diagnosis of Energy Inefficiency for Smartphone Applications	389
18.1	Introduction	390
18.2	Background	392
18.3	Empirical Study	393
18.3.1	Problem Magnitude	394
18.3.2	Diagnosis and Bug-Fixing Efforts	394
18.3.3	Common Patterns of Energy Bugs	396
18.3.4	Threats to Validity	402
18.4	Energy Efficiency Diagnosis	402
18.4.1	Approach Overview	402
18.4.2	Application Execution and State Exploration	403
18.4.3	Detecting Missing Sensor or Wake Lock Deactivation	406
18.4.4	Sensory Data Utilization Analysis	407
18.5	Experimental Evaluation	413
18.5.1	Experimental Setup	414
18.5.2	Effectiveness and Efficiency	416
18.5.3	Necessity and Usefulness of AEM Model	422
18.5.4	Impact of Event Sequence Length Limit	424
18.5.5	Comparison with Resource Leak Detection Work	427
18.5.6	Energy Saving Case Study	428
18.5.7	Discussions	429
18.6	Related Work	432
18.6.1	Energy Efficiency Analysis	432
18.6.2	Energy Consumption Estimation	433
18.6.3	Resource Leak Detection	434
18.6.4	Information Flow Tracking	435
18.7	Concluding Remarks	435
	References	435
Part VI Conclusion and Future Outlook		
19	Conclusion and Future Outlook	441

Internetware

A New Software Paradigm for Internet Computing

Mei, H.; Lü, J.

2016, XXVIII, 442 p. 191 illus., 117 illus. in color.,

Hardcover

ISBN: 978-981-10-2545-7