

Doug Rosenberg • Barry Boehm • Matt Stephens
Charles Suscheck • Shobha Rani Dhalipathi
Bo Wang

Parallel Agile – faster delivery, fewer defects, lower cost



Springer

Contents

1	Parallel Agile Concepts	1
1.1	Partitioning a System for Parallel Development	1
1.2	Just Enough Planning	3
1.3	Feedback-Driven Management, Model-Driven Design	4
1.4	Risk Management	5
1.5	Project Management	6
1.6	Executable Domain Models	7
1.7	Parallel Agile Process	9
1.8	Scalability and Evolution of Parallel Agile from ICONIX Process	12
1.9	Summary	13
	References	14
2	Inside Parallel Agile	15
2.1	Code First, Design Later	15
2.2	Prototyping as Requirements Exploration	16
2.3	Overview of the Parallel Agile Process	16
2.4	Inception	17
2.4.1	Evolving Database Schemas	17
2.4.2	Enabling Integration Between Developers	18
2.5	Parallel Development Proceeds in Three Phases After Inception	19
2.6	Proof of Concept (Building the Right System)	19
2.7	Minimum Viable Product (Building the System Right)	20
2.7.1	Using MVC Decomposition to Make Your Use Cases Less Ambiguous	21
2.7.2	Using Parts of Parallel Agile with Scrum	21
2.7.3	Adding Controllers to the Scrum Backlog	22
2.7.4	Tracking Agile Projects by Epic, User Story, and Task	23
2.8	Optimization and Acceptance Testing	23

2.9	Balancing Agility and Discipline	24
2.10	Summary	25
	References.	26
3	CodeBots: From Domain Model to Executable Architecture	27
3.1	Solving Problems to Enable Parallelism	28
3.1.1	Parallel Agile Versus Agile/ICONIX	29
3.1.2	Cost Benefits	30
3.1.3	Origin of Executable Architectures	30
3.1.4	Developer to Developer Integration	31
3.1.5	Resilience to Staff Turnover.	31
3.2	Domain-Driven Prototyping.	32
3.2.1	Introducing CodeBot	34
3.2.2	What Is Prototyping?	35
3.2.3	CarmaCam Domain Modeling Workshop	36
3.2.4	The Prototyping Is Done—What’s Next?	45
3.3	Using CodeBot During the MVP Phase.	46
3.3.1	What to Expect from Your UML Relationships.	47
3.4	Deployment Architecture Blueprints (Preview).	49
3.5	Summary	49
4	Parallel Agile by Example: CarmaCam	53
4.1	CarmaCam Architecture.	55
4.2	Sprint 1: Proof of Concept	56
4.2.1	Executable Domain Model.	59
4.2.2	Use Cases and Storyboards	60
4.2.3	Prototypes.	62
4.3	Sprint 2: Minimum Viable Product	68
4.3.1	MVC Decomposition.	68
4.3.2	State Transition Diagram	70
4.3.3	Testing	70
4.3.4	Server Migration to AWS.	74
4.4	Sprint 3: Optimization	74
4.4.1	Web App.	75
4.4.2	Mobile App.	76
4.4.3	Emergency Alert Receiver App	77
4.4.4	Server App	79
4.5	What’s Next?	79
4.6	Summary	80
5	Taking the Scream Out of Scrum	81
5.1	Agile Mindset.	82
5.1.1	General Agile Mindset Misconceptions.	85
5.1.2	The Sweet Spot of Parallel Agile in the Agile Mindset	85
5.1.3	Scaled Agile Framework and Parallel Agile.	86

5.2	Scrum as It Should Be: A Quick Overview	87
5.2.1	Misconceptions About Scrum Roles	89
5.2.2	Misconceptions About Scrum Events	90
5.2.3	Misconceptions About Scrum Artifacts	93
5.3	Example: Parallel Agile with Backlogs and a Small Team	95
5.3.1	From Product Vision to Product Delivery	96
5.3.2	Preparing the Product Backlog	96
5.3.3	Sprint Planning	98
5.3.4	Sprint	102
5.3.5	Sprint Review	104
5.4	Summary	105
	References	106
6	Test Early, Test Often	107
6.1	A Note About Software Quality	109
6.2	Errors of Commission vs. Errors of Omission	110
6.3	Acceptance Testing Fails When Edge Cases Are Missed	112
6.4	CarmaCam Example	113
6.5	Testing at Different Levels	114
6.6	A Capsule Summary of Domain Oriented Testing	115
6.7	Drive Unit Tests from the Code	117
6.8	Drive Component Tests from the Use Case Scenarios	118
6.8.1	When Should Component Tests Be Written?	119
6.9	Drive Acceptance Tests from the User Stories and Use Case Scenarios	120
6.9.1	Who Is Responsible for the Acceptance Tests?	121
6.9.2	Manual vs. Automated Acceptance Tests	122
6.9.3	Acceptance Test Early and Often	123
6.9.4	Creating a Manual Acceptance Test Script	124
6.9.5	Creating an Automated Acceptance Test with BDD and Cucumber	125
6.9.6	Creating an Automated Acceptance Test with DDT	126
6.9.7	Prototype to Discover Requirements, Review Them Carefully, and Test Each One	127
6.10	Summary	129
	References	130
7	Managing Parallelism: Faster Delivery, Fewer Defects, Lower Cost	131
7.1	Believe in Parallelism: Resistance Is Futile	133
7.1.1	Multicore CPUs	135
7.1.2	Elastic Cloud of Developers	135
7.1.3	You Want It WHEN?	137
7.2	Managing Parallelism: Instant Tactical Assessment	138
7.2.1	Task Estimation from Sprint Plans	140
7.2.2	Rapid and Adaptive Planning	140

7.2.3	Seizing the Initiative	141
7.2.4	Analyzing Use Case Complexity	143
7.2.5	Redeploying Resources as the Situation Evolves	145
7.2.6	Accelerating a Late Project by Adding Developers	145
7.3	Improving Quality While Going Faster	147
7.3.1	Generating Database Access Code	147
7.3.2	Generating Acceptance Tests from Use Cases	147
7.3.3	Testing in Parallel with Developing	148
7.3.4	Hurry, but Don't Rush	149
7.4	Lowering Development Cost	150
7.4.1	Doing "Just Enough" Design Reduces Costs	150
7.4.2	Combining Planning with Feedback	151
7.5	Summary	151
	References	152
8	Large-Scale Parallel Development	153
8.1	Parallel Agile and the Incremental Commitment Spiral Model	155
8.1.1	ICSM Phases Map to Proof of Concept, Minimum Viable Product, and Optimization	156
8.1.2	ICSM Spiral Includes Prototyping, Design, and Acceptance Testing	156
8.1.3	ICSM Principles Were Developed on Very Large Projects	158
8.2	Parallel Agile Critical Success Factors	158
8.3	TRW-USAF/ESC CCPDS-R Parallel Development	160
8.3.1	CCPDS-R Evidence-Based Decision Milestones	161
8.3.2	CCPDS-R Parallel Development	162
8.4	Parallel Development of Even Larger TRW Defense Software Systems	162
8.5	Comparing the Parallel Agile Approach and CSFs and Other Successful Scalable Agile Approaches	164
8.5.1	The Speed, Data, and Ecosystems Approach	165
8.5.2	The Scaled Agile Framework (SAFe) Approach	167
8.6	Conclusions and Latest Parallel Agile Scalability Experience	167
8.7	Summary	168
	References	168
9	Parallel Agile for Machine Learning	169
9.1	Phase 1: Proof of Concept and Initial Sprint Plan	170
9.1.1	CarmaCam Incident Reports	172
9.1.2	CarmaCam Emergency Alert Videos	173
9.1.3	Identifying Multiple Lane Changes at High Speed	175
9.1.4	Training Machine Learning Models from CarmaCam Videos	177

9.1.5	Training Machine Learning Models Using Video Games	179
9.1.6	Phase 1 Results and Summary	180
9.2	Phase 2: Minimum Viable Product – Detecting a Likely DUI from Video	182
9.3	Summary	184
	Reference	184
Appendix A:	The Scream Guide	185
Appendix B:	Architecture Blueprints	205
References	215
Index.	217