1. Record Nr. UNISA996464533903316

Titolo	Domain-specific languages in practice : with JetBrains MPS / / Antonio Bucchiarone [and three others], editors
Pubbl/distr/stampa	Cham, Switzerland : , : Springer, , [2021] ©2021
ISBN	3-030-73758-6
Descrizione fisica	1 online resource (342 pages)
Disciplina	005.11
Soggetti	Domain-specific programming languages
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di bibliografia	Includes bibliographical references.
Nota di contenuto	Intro -- Preface -- MPS in Industrial Applications -- MPS in Research Projects -- Teaching and Learning with MPS -- Perspectives -- Acknowledgments -- References -- Contents -- JetBrains MPS: Why Modern Language Workbenches Matter -- 1 Introduction to the Domain of Language Workbenches -- 1.1 A Brief History of the MPS Project -- 1.2 The Business Value of Language Workbenches -- 1.2.1 Productivity -- 1.2.2 Quality -- 1.2.3 Leveraging Expertise -- 2 MPS Terminology and Notations -- 2.1 Abstract Syntax Tree -- 2.2 Node -- 2.3 Concept -- 2.4 Models vs. Meta-models -- 2.5 Language -- 2.6 Modules -- 2.7 Models -- 2.8 Generator -- 3 BaseLanguage -- 4 Projectional Editor -- 4.1 Notations -- 4.2 Benefits of Projectional Editing -- 4.3 Notations Trade-Offs -- 4.4 Reflective Editor -- 5 The MPS Way of Defining Languages -- 5.1 Structure -- 5.2 Editor -- 5.3 Constraints -- 5.4 Typesystem -- 5.4.1 Type Calculation -- 5.4.2 Static Code Analysis -- 5.5 Generator -- 6 Integration with Other Systems -- 6.1 Language Plugins -- 6.2 Standalone IDEs -- 6.3 Build Language -- 6.4 Persistence -- 6.5 Version Control -- 6.6 Third-Party Tooling -- 7 Language Versioning and Migrations -- 8 Testing Language Definitions -- 8.1 Debugging -- 8.2 Editor Tests -- 8.3 Node Tests -- 8.4 Migration Tests -- 8.5 Generator Tests -- 9 The MPS Community -- 9.1 Sources of Information -- 9.2 MPS Extensions -- 9.3 Language Repository -- 10 Conclusion and Future Developments -- References -- Part I JetBrains MPS in Industrial Applications -- Use MPS to Unleash

the Creativity of Domain Experts: Language Engineering Is a Key Enabler for Bringing Innovation in Industry -- 1 Introduction -- 1.1 Disconnect Between Domain Experts and Tool Providers -- 1.2 Intended Audience -- 1.3 Structure of This Chapter -- 2 Modeling Failures in Automated Production Lines.
2.1 Ensuring Product Quality for Smart Automated Production -- 2.1.1 Model-Based Approach to pFMEA -- 2.1.2 Meta-Model for Automated pFMEA -- 2.2 Example: Assembling a Quadrocopter in a Smart Factory -- 2.3 Conclusion -- 3 Modeling Contractual Agreements -- 3.1 ADORA: DSLs for Modeling Contractual Agreements -- 3.1.1 Modeling Economic Aspects -- 3.1.2 Integrating Technical Aspects -- 3.2 Evolution of ADORA -- 4 Specifying Parameters of Computed Tomography Scanners Product Lines -- 4.1 Somaris DSL: Specification of CT Parameter Configurations -- 4.2 Project Timeline -- 4.2.1 Language Engineering -- 4.2.2 Involving Domain Experts -- 4.2.3 Somaris DSL in Numbers -- 4.3 Examples of Models -- 4.4 Development Infrastructure and Process -- 4.4.1 Repository and Build Infrastructure -- 4.4.2 Language Evolution and Migration of Model Instances -- 4.4.3 Validation and Verification -- 4.5 Conclusion and Outlook -- 5 Lessons Learnt -- 5.1 Adoption -- 5.2 Tooling-Driven Research -- 5.3 Mostly Used MPS Features -- 6 Conclusions -- References -- JetBrains MPS as Core DSL Technology for Developing Professional Digital Printers -- 1 Introduction -- 2 Product Variability -- 2.1 Variability at Canon Production Printing -- 2.1.1 Designing for Variability -- 2.1.2 Development Effort and User Base -- 2.2 Product Line Engineering -- 2.2.1 Solution Outline -- 2.2.2 Example -- 2.3 Advantages of MPS -- 2.4 Shortcomings of MPS -- 2.5 Status and Outlook for Product Variability Modeling -- 3 Mechanics-Software Interface -- 3.1 Paper Paths -- 3.2 Parts -- 3.3 HappyFlow Sheet Timing -- 3.4 Sheet Scheduling Constraints and Verification -- 3.5 Product Variants and Modularity -- 3.6 Continuous Integration of Model-Based Design Artifacts -- 3.7 Concluding Remarks on MSI -- 4 Hardware-Software Interface -- 4.1 Initial Version of HSI -- 4.2 Combined Domains and Domain-Specific Editing.
4.3 Harnessing Modularization -- 4.4 Coping with Increased Complexity -- 4.5 Status and Outlook for HSI -- 5 Domain-Specific State Machine Specification -- 5.1 Yet Another State Machine Language -- 5.2 Open Interaction Language (OIL) -- 5.3 Prototype Tool -- 5.4 Behavioral Verification -- 5.5 Usability and Maintainability -- 5.6 Conclusion for OIL -- 6 Virtual Printer Configuration -- 6.1 Specifying Simulation Configurations -- 6.2 Example Virtual Printer Configuration DSLs -- 6.2.1 Print Head Specifications -- 6.2.2 Carriage Motion Specification -- 6.3 Outlook for Virtual Printer Configuration -- 7 Collaborative Domain-Specific Modeling -- 7.1 Blended Collaborative Domain-Specific Modeling -- 7.2 DSL Widgets in Custom GUIs -- 7.3 Outlook for Collaborative Modeling -- 8 Conclusions and Lessons Learned -- References -- A Domain-Specific Language for Payroll Calculations: An Experience Report from DATEV -- 1 Introduction -- 2 Terminology -- 3 Context -- 3.1 Business Context -- 3.2 Business Challenges -- 3.3 The Legacy System -- 3.4 Why a DSL -- 3.5 Language Development -- 3.6 System Architecture -- 3.7 Current Status of the System -- 4 Case Study Setup -- 4.1 Research Questions -- 4.2 Data Collected -- 5 The Language and Tooling -- 5.1 A Functional Language -- 5.2 Execution -- 5.3 High-Level Structure -- 5.4 Domain Data Types -- 5.5 Tables -- 5.6 Temporal Data -- 5.7 Indexing -- 5.8 Declarative Dependencies -- 5.9 Versioning -- 5.10 Data Validation -- 5.11 Testing Support -- 5.12 External Components -- 5.13 IDE Features -- 6 Evaluation -- 6.1 RQ1 Is a Suitably Designed

DSL Able to Significantly Reduce the Perceived Complexity in the Payroll Domain? -- 6.2 RQ2 Does the Use of DSLs and the Associated Tools Increase or Decrease the Quality of the Final Product?.
6.3 RQ3 Can a DSL that Reduces Complexity be Taught to Domain Experts in a Reasonable Amount of Time? -- 6.4 RQ4 How Well Does the DSL and Its Use for Application Development Fit with Established IT Development Processes and System Architecture? -- 7 General Learnings -- 8 Validity -- 9 Related Work -- 10 Conclusions -- References -- FASTEN: An Extensible Platform to Experiment with Rigorous Modeling of Safety-Critical Systems -- 1 Introduction -- 2 The FASTEN Platform -- 3 Modeling of Requirements -- 4 Formalizing System-Level Designs with SMV-Based DSLs -- 5 Modeling Safety Aspects -- 5.1 Modeling Support for Hazards and Risk Analysis -- 5.2 Modeling Safety Cases -- 6 Experiences from the Automotive Domain -- 6.1 System Model Specification in SysML -- 6.2 Contract Specification Patterns -- 6.3 Case Studies -- 7 Discussion and Lessons Learned -- 7.1 Discussion -- 7.2 MPS Features Supporting Our Work -- 7.3 Open Challenges with MPS-Based Tooling -- 8 Conclusions and Future Work -- References -- Migrating Insurance Calculation Rule Descriptions from Word to MPS -- 1 Introduction -- 2 Project Description and Challenges -- 2.1 Data Model Definition with VADM -- 2.2 Functional Specification in FuMo -- 2.2.1 Description -- 2.2.2 Challenges -- 2.3 Executable Implementation in C -- 2.3.1 Description -- 2.3.2 Challenges -- 2.4 Functional and Regression Tests -- 2.4.1 Description -- 2.4.2 Challenges -- 2.5 IT Department Team Authoring VADM and FuMo -- 2.5.1 Description -- 2.5.2 Challenges -- 2.6 External Service Provider Implementing Executable C and Tests -- 2.6.1 Description -- 2.6.2 Challenges -- 3 Proposed Solution -- 3.1 Solution Technologies -- 3.2 VADM -- 3.2.1 Language Design -- 3.2.2 Language Implementation -- 3.3 FuMo -- 3.3.1 Language Design -- 3.3.2 Language Implementation -- 4 Evaluation and Lessons Learned -- 4.1 Language Implementation.
4.2 Import and Generation -- 4.2.1 Import Source -- 4.2.2 Big Bang vs. Incremental Transformation -- 4.2.3 Cleaning Up Sources -- 4.2.4 Lifting from C to FuMo DSL -- 4.2.5 Handling VADM Access -- 4.3 First Importing Approach -- 4.4 Second Importing Approach -- 4.5 Analyzing Test Failures at Scale -- 5 Conclusion -- 5.1 Technical Advantages and Shortcomings -- 5.2 Project Results -- References -- Part II JetBrains MPS in Research Projects -- Projecting Textual Languages -- 1 Introduction -- 2 Motivation -- 3 Background -- 3.1 Software Language Engineering -- 3.2 Syntax of Textual and Projectional Languages -- 4 Approach: Projecting Textual Languages -- 4.1 Mapping Grammars to Concept Hierarchies -- 4.2 Mapping Grammars to Editor Aspects -- 4.3 Editor Improvement: AST Pruning -- 4.4 Translating Textual Programs into Projectional Models -- 4.5 Architecture -- 5 Case Study -- 5.1 Language Description -- 5.2 Editor Aspect -- 5.3 Program's Usability -- 5.4 Discussion -- 6 Limitations -- 7 Related Work -- 7.1 Grammar to Model -- 7.2 Editor Generation -- 8 Conclusions and Future Work -- References -- Engineering Gameful Applications with MPS -- 1 Introduction -- 2 Motivations and Contribution -- 2.1 MDA -- 2.2 Elemental Tetrad -- 2.3 Open Issues and Contribution -- 3 Case Study: PapyGame Design with GDF -- 4 Engineering the Gamification Design Framework (GDF) with MPS -- 4.1 MPS Projectional Editors -- 4.2 MPS Generators -- 5 Lessons Learned and Future Investigations -- 6 Conclusions -- References -- Learning Data Analysis with MetaR -- 1 Introduction -- 2 Domain -- 2.1 What Is Data Analysis? -- 2.2 Which Data? -- 2.3 The R Statistical Language -- 2.4 MetaR Languages -- 2.5 Relation with R -- 2.5.1 Programming

Paradigm -- 2.5.2 External Packages -- 3 Development and User Community -- 3.1 Development -- 3.2 Target Audience -- 3.3 User Community.
4 Requirements, Design, and Architecture.