Systematic Metric Systems Engineering: Reference Architecture and Process Model

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften
der RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Matthias Vianden
aus
Aachen

Berichter: Prof. Dr. rer. nat. Horst Lichter
Prof. Dr. rer. nat. Stefan Wagner

Tag der mündlichen Prüfung: 20.07.2016

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 26

Matthias Vianden
RWTH Aachen University

Systematic Metric Systems Engineering

Reference Architecture and Process Model

Shaker Verlag
Aachen 2017

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: D 82 (Diss. RWTH Aachen University, 2016)

Copyright Shaker Verlag 2017
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-5111-7
ISSN 1869-9170

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • e-mail: info@shaker.de

Abstract

In the recent past, the research community contributed a considerable amount of work to extend the understanding of the theoretical foundations of metric systems. However, a dedicated approach for engineering of metric systems is still missing. As a result, they are often developed chaotically. This thesis introduces MeDIC – a dedicated metric systems engineering approach, which fills this gap. MeDIC supports flexible conception, design, construction, and operation of metric systems. The approach is based on two pillars: the *MeDIC process model* and the *MeDIC reference architecture*. They integrate software engineering best practices, emerging concepts, and well-established metric-related standards and techniques. The MeDIC reference architecture provides technical guides with a layered architecture blue-print of loosely interconnected micro-services. The MeDIC process model provides ready-to-use process elements and artifacts (fragments), which drastically ease the setup of a specific engineering process. The reference architecture and process model are based on formal foundations, which provide additional benefits for conceptual analysis of metrics systems. Various field studies, in cooperation with multiple industry partners, were used to evaluate the approach. This thesis provides insight into three selected field studies, which utilize various aspects of MeDIC in industrial environments. The evaluation shows the practical application, usefulness, and efficiency of MeDIC. Challenges associated with the development and operation of metric systems in industrial environments can thus be overcome by MeDIC. As a result, the engineering of better, more reliable, and sustainable metric systems is possible.

Kurzfassung

In der Vergangenheit wurde vermehrt an den theoretischen Grundlagen der (Software-) Metriken gearbeitet. Das allgemeine Verständnis hat sich seitdem stark weiterentwickelt und neben den theoretischen Grundlagen hat sich auch das Verständnis des Metrikmanagement weiterentwickelt. Hieraus entwickelte sich ein etablierter Stand der Technik, welcher sich in diversen Standards widerspiegelt (ISO 15939, CMMI MA). Es fehlt allerdings immer noch ein spezieller Engineering-Ansatz für Metriksysteme und die damit verbundene Messinfrastruktur, was dazu führt, dass diese oft chaotisch entwickelt werden. Der in dieser Arbeit vorgestellte flexible Engineering-Ansatz MeDIC schließt diese Lücke und adressiert sowohl die klassischen Phasen des Software-Engineerings: Konzeption, Entwurf und Konstruktion, als auch Betrieb von Metriksystemen. MeDIC besteht aus zwei fundamentalen Teilen: dem MeDIC-Prozessmodell und der MeDIC-Referenzarchitektur. Diese integrieren best-practices der Software-Entwicklung, moderne Konzepte und etablierte Techniken und Standards im Bereich der Metriken miteinander. Die MeDIC-Referenzarchitektur stellt eine Blaupause einer geschichteten Architektur von lose gekoppelten Micro-Services zur Verfügung und erhöht damit das Verständnis der technischen Konzepte. Das MeDIC-Prozessmodell enthält fertig verwendbare Prozessbausteine und Artefakte (-Fragmente), welche den Aufbau eines dedizierten Engineering-Prozesses drastisch vereinfachen. Die Referenzarchitektur und das Prozessmodell sind mit einer formalen Basis untermauert, welche zusätzlich die Analyse von Metriksystemen auf einem theoretischen und konzeptuellen Niveau ermöglicht. MeDIC wurde in zahlreichen industriellen Feldstudien evaluiert. Diese Arbeit stellt drei ausgewählte Feldstudien vor, welche unterschiedliche Aspekte von MeDIC im industriellen Umfeld verwenden. Hierdurch wird die praktische Anwendbarkeit, Nützlichkeit und Effektivität des Ansatzes gezeigt. MeDIC hilft, viele der praktischen Probleme beim Entwickeln und Betreiben von Metriksystemen im industriellen Umfeld zu überwinden. In der Zukunft können diese Systeme mit Hilfe von MeDIC demnach besser, zuverlässiger und nachhaltiger entwickelt werden.

Contents

I. Introduction and Foundations 1

1. Introduction 3
1.1. Metric Systems Engineering Challenges . . . 6
1.1.1. Large Software Development Companies . . . 6
1.1.2. Small and Medium Software Development Companies . . . 7
1.1.3. Main Challenges . . . 9
1.1.4. Summary . . . 10
1.2. Top-Level Requirements . . . 12
1.2.1. Literature Analysis . . . 13
1.2.2. Summary . . . 16
1.3. Research Questions and Contribution . . . 17
1.3.1. Contribution . . . 17
1.4. Research Field and Central Related Work . . . 19
1.4.1. Service-Oriented Measurement Infrastructures . . . 19
1.4.2. Software Project Control Centers . . . 22
1.4.3. Summary and Conclusion . . . 25
1.5. MeDIC - A Metric Systems Engineering Approach . . . 27
1.5.1. Flexibility . . . 29
1.5.2. Information Need Driven . . . 30
1.5.3. Usable Metric Systems . . . 32
1.6. Summary . . . 33

2. Conceptual Foundations 35
2.1. Metric Portfolio . . . 36
2.1.1. Metric Terminology . . . 36
2.1.2. Metrics System Dynamics and Measurement Data Flow . . . 38
2.1.3. Summary . . . 40
2.2. Metric Reuse . . . 41
2.2.1. Metric Reuse Dimensions . . . 41
2.2.2. Metric Reuse in the Literature . . . 42
2.2.3. Metric Reuse by Metric Variability . . . 43
2.2.4. Formal Foundation to Metric Variability . . . 44
2.2.5. Summary . . . 45
2.3. Formal Foundation to Metric System Dynamics . . . 46
2.3.1. Related Work . . . 46
2.3.2. Overview . . . 47
2.3.3. Preface . . . 48
2.3.4. Measurement Data and Measurements . . . 49
2.3.5. Compatibility . . . 51
2.3.6. Satisfiability . . . 55

2.3.7. Measurement Producer 56
2.3.8. Calculation Termination 60
2.4. Summary 62

II. MeDIC Reference Architecture 63

3. Introduction, Requirements and Foundations 65
3.1. Design Foundations and Reference Architecture Requirements 67
3.1.1. Polylithic Micro Service-based Measurement Infrastructures 67
3.1.2. Specific Requirements 68
3.1.3. Reference Architecture Requirements Summary 73
3.2. The API Specification Language 74
3.3. Integration Architecture Alternatives 77

4. Logical Reference Architecture and Physical System View 79
4.1. Logical Reference Architecture 79
4.2. Physical System View 83
4.2.1. Data Provider Systems 83
4.2.2. Support Systems 84
4.2.3. Core Systems 85

5. Technical Reference Architecture 87
5.1. Overview 88
5.1.1. Measurement Data Flow 90
5.1.2. Concept to Implementation Mapping 92
5.1.3. Discussion and Design Alternatives 92
5.2. Data Transport and Integration 97
5.2.1. Enterprise Measurement Data Bus (EMDB) 98
5.2.2. EMDB Messages 99
5.2.3. Integration and Reuse 100
5.2.4. Important EMDB Services 101
5.2.5. Additional Service Topics 107
5.2.6. Summary 107
5.3. Calculation Access 109
5.3.1. Design Decisions and Related Work 111
5.3.2. EUrEKA Indicator Access APIs 112
5.3.3. EUrEKA Kernel Description Meta Model 114
5.3.4. EUrEKA Registry 118
5.3.5. EUrEKA Producer Gateway (optional) 120
5.3.6. EUrEKA Consumer 122
5.3.7. EUrEKA Indicator Wrapper (optional) 123
5.3.8. Summary 125

5.4. Data Adapter Reference Architecture 127
5.4.1. Adaption Patterns and Dynamic View 127
5.4.2. Static Reference Architecture 134
5.4.3. Summary 136
5.5. Metric Kernel Reference Architecture 137
5.5.1. Design Alternatives 138
5.5.2. Dynamic View 141
5.5.3. Summary 143
5.6. Visualization Reference Architecture 145
5.6.1. Metric-based Monitoring Dashboards 146
5.6.2. Visualization Frontend Classification 146
5.6.3. Component View 147
5.6.4. Dynamic View 148
5.6.5. Summary 151
5.7. Technical Integration of Operation Services 152
5.8. Summary of the Technical Reference Architecture 154

6. Operation Systems and Services 155
6.1. Monitoring System 156
6.1.1. Information Needs Satisfied by the Monitoring System 156
6.1.2. Monitoring System Reference Architecture 162
6.1.3. Monitoring System Summary 167
6.2. Logging System 169
6.2.1. Information Needs Satisfied by the Logging System 169
6.2.2. Logging System Reference Architecture 171
6.2.3. Logging System Summary 172
6.3. Lookup System (optional) 173
6.3.1. Use Cases and Requirements for the Lookup System 173
6.3.2. Directory System Reference Architecture 174

7. MeDIC Reference Architecture Formalisms 179
7.1. Formalism for Service States 180
7.1.1. Maintenance 180
7.1.2. Sync with Data Provider 181
7.2. Formal Basis of the Technical Reference Architecture 182
7.2.1. Preface 182
7.2.2. Measurement Messages 183
7.2.3. Data Adapter 184
7.2.4. Metric Kernel 185
7.2.5. Data Processing in an EMI 189
7.2.6. Formalism Summary 191
7.3. Formalism Example: Ticket Statistics 192
7.3.1. Introduction and Definition of the EMI 192
7.3.2. Metric Definition 192

7.3.3. Metric Kernel: Measurement Consumer 195
7.3.4. Metric Kernel: Data Storage 195
7.3.5. Correctness Proof of the Storage Function 197
7.3.6. Metric Kernel: Measurement Producer 200
7.3.7. Termination Proof of the Kernel and the EMI 200
7.3.8. Example Summary 201

III. MeDIC Process Model 203

8. Process Model Foundations 205
8.1. Process Environment Assumptions 207

9. The Metric System Engineering Process Model 209
9.1. Process Model Core 210
9.2. Process Overview 213
9.3. Roles 215
9.3.1. Metric Customer 215
9.3.2. Metric Expert 216
9.3.3. Architect 218
9.3.4. Developer 219
9.3.5. Operator 220
9.3.6. Role Involvement 222
9.4. Process Initialization 224

10. The Conception Phase 227
10.1. Requirements Gathering 229
10.1.1. Activity Overview 229
10.1.2. Plan Requirements Gathering and Information Need identification 230
10.1.3. Execute RE Plan 231
10.1.4. Process Results 233
10.2. Prototype and Evaluate 234
10.2.1. Consolidate Info Needs 234
10.2.2. Design Monitors, Design Metrics and Prepare Prototypes 235
10.2.3. Evaluate With Metric Customers 236
10.3. Plan Increment 238
10.3.1. Integrate Information Needs and Design Logical Architecture . . . 238
10.3.2. Review and Prioritize Increment Plan 239
10.3.3. Finish Increment Planing 239
10.4. Conception Summary 240

11. The Design Phase 241
11.1. Identify Metric Services 242
11.1.1. Setup the Design Plan and Design Document 243

11.2. Design and Evaluate 244
11.2.1. Design Services and Integration 244
11.2.2. Design Metric Service Tests 245
11.2.3. Evaluate design 245

12. The Construction and the Operation Phase 247
12.1. The Construction Phase 247
12.2. The Operation Phase 249
12.2.1. Deploy and Setup a new Metric Kernel 249
12.2.2. Best Practices for Handling Common Errors and Exceptions . . . 249
12.2.3. Triggering a new Iteration 255
12.3. Summary 256

IV. Evaluation, Tool Support, and Lessons Learned 257

13. Evaluation by Selected Field Studies 259
13.1. Project Risk Metric System for a Large IT Service Provider 260
13.1.1. Process 261
13.1.2. Risk Metrics Architecture 265
13.1.3. Experience and Best Practices 268
13.2. Software Project Metrics System for SSE Lab 271
13.2.1. Process 271
13.2.2. Architecture 273
13.2.3. Experience 275
13.3. Flow-based Visual Ticket Analysis 277
13.3.1. Key Concepts 277
13.3.2. Architecture - First Version 280
13.3.3. Architecture - Second Version 282
13.3.4. Experience 286

14. Tooling 289
14.1. MeDIC Metric Documentation Tools 290
14.2. MeDIC Metric Management Support Tool 293
14.3. MeDIC Dashboard and SCREEN 296
14.3.1. Architecture - MeDIC Dashboard 298
14.3.2. Architecture - SCREEN 299
14.4. EMI Services 301
14.4.1. EMS - EMI Monitoring Service 301
14.4.2. ELS - EMI Logging Service 302
14.4.3. EDS - EMI Directory Service 303
14.4.4. ERS - EMI Render Service 303
14.5. EMI Framework 306

15. Lessons Learned and Discussion 309
15.1. Security 309
15.2. Flexibility 311
15.3. Ease-of-Use 311
15.3.1. Ease-of-Use of the Reference Architecture 311
15.3.2. Ease-of-Use of the Process Model 312
15.4. Effectiveness 312
15.4.1. Effectiveness of the Reference Architecture 312
15.4.2. Effectiveness of the Process Model 313
15.5. Efficiency 313

V. Conclusion and Future Work 315

16. Conclusion and Future Work 317
16.1. Future Work 317
16.2. Conclusion 318

VI. Appendix 321

A. Symbol Lists 323
A.1. Symbols used in the Foundation Formalism 323
A.2. Symbols used in the Reference Architecture Formalism 325

B. Process Guides, Checklists, and Document Descriptions for the Process Model327
B.1. Conception Phase 327
B.1.1. Information Need Gathering – Guidelines for the Execution 327
B.1.2. Plan Increment – Guidelines for Coherent Increments 329
B.2. Design Phase 330
B.2.1. Services Reuse Decision Aid – Checklist 330
B.2.2. Design Guides for EMI Services 332
B.2.3. Design Guide for Test Selection and Test Stage Description 334
B.3. The Design Document 336
B.3.1. Rough Design of the Complete Metric Application 336
B.3.2. Exception Behavior 336
B.3.3. Fine Design of the Integration 337
B.3.4. Fine Design of each Services 337
B.3.5. Tests 339

C. Student Theses in the Context of this Thesis 341
C.1. Diploma Theses 341
C.2. Master Theses 341
C.3. Bachelor Theses 342

List of Figures

1.1. Metric system decomposition 4
1.2. Development gap in metric systems engineering. 11
1.3. The main parts of MeDIC 27
1.4. MeDIC metric system engineering: Overview 28

2.1. Static relations between metric portfolio terms 37
2.2. Metric dynamics of a metric system by the means of measurement producers and measurement consumers 38
2.3. Metric dynamics and measurement data flow example 39
2.4. Conceptual model of the data flow in a metric system 39
2.5. Static view on central metric variability concepts 44
2.6. Overview of the central parts to our formal foundations to metric system dynamics 48
2.7. Measurement producer connecting metric, measurement, and the variability model 56

4.1. Model for the logical reference architecture as UML class diagram 80
4.2. Example for the logical architecture and logical decomposition of an enterprise measurement infrastructure 81
4.3. System View 83

5.1. Technical layers and services in an enterprise measurement infrastructure 88
5.2. Simplified data and control flow in an EMI 91
5.3. Metric concepts and their corresponding implementations in an enterprise measurement infrastructure 93
5.4. Zoom into the data transport and integration layer of the MeDIC reference architecture 97
5.5. Publish/subscribe topics inside the EMDB 98
5.6. Example for a EMDB message specialization hierarchy and relation to EMDB topics 100
5.7. Integration between Data Adapter and Metric Kernel via EMDB Messages 100
5.8. Reusing a general metric kernel with specific EMDB messages 101
5.9. Message Gateway internal component view 102
5.10. Message Cache internal component view 104
5.11. Zoom into the calculation access layer of the MeDIC reference architecture 109
5.12. Overview and layers of the enterprise uniform metric kernel access (EUrEKA) design 110
5.13. EUrEKA metric kernel description meta model as UML class diagram . . 115
5.14. Example for a risk matrix that can be feed by the risk data type from the previous example 117
5.15. EUrEKA registry service internal component view 118

5.16. EUrEKA overview using the EUrEKA producer gateway service 120
5.17. EUrEKA producer gateway service internal component view 121
5.18. EUrEKA consumer internal component view 122
5.19. Wrapper Configuration Model and Wrapper production 123
5.20. EUrEKA indicator wrapper service internal component view 124
5.21. Zoom into the data adapter layer of the MeDIC reference architecture . . 127
5.22. Icons for the different data adapter pattern 127
5.23. UML sequence diagram for the concept of the push-forward adapter pattern 128
5.24. UML sequence diagram for the concept of the pull-forward adapter pattern 129
5.25. UML sequence diagram for the concept of the invoke-pull adapter pattern 130
5.26. UML sequence diagram for the concept of the invoke-dump adapter pattern 132
5.27. Static reference architecture for Data Gateways 134
5.28. Static reference architecture for pull-based data adapter 135
5.29. Zoom into the metric kernel related layers of the MeDIC reference architecture 137
5.30. Static monolithic metric kernel reference architecture. 138
5.31. Static metric kernel reference architecture with separated components. . . 140
5.32. EMDB Message Processing 141
5.33. UML sequence diagram for the behavior of the metric kernel components when data is requested via an indicator access API 143
5.34. Zoom into the visualization layer of the MeDIC reference architecture . . 145
5.35. Dashboard Application - Component View 148
5.36. Typical M^2 dashboard visualization dynamic as UML sequence diagram . 150
5.37. Typical analysis tool visualization dynamic as UML sequence diagram . . 151
5.38. Integration in an Operation System between the Operation Service and EMI Services 152
5.39. Operation System Component View 153

6.1. Static reference architecture for the monitoring client agent 163
6.2. UML activity diagram for the behavior of the monitoring service and monitoring client agent during an alive-check. 165
6.3. UML sequence diagram for the production of the performance indicators in a monitoring client agent. 167
6.4. Static reference architecture for the logging client agent 171
6.5. Static reference architecture for the directory system 175

8.1. Scope of the metric systems engineering process model 205

9.1. Metric Systems Engineering Process: phases, increments, and iterations . 210
9.2. MeDIC engineering process model: phase details and core activities 213
9.3. Workload of the roles in the different phases of the development process . 222

10.1. Conception phase overview 227
10.2. Requirements Gathering activity as BPMN diagram 230
10.3. Prototype and evaluate activity as BPMN diagram 234

10.4. Plan Increment activity as BPMN diagram 238

11.1. Design phase overview as BPMN diagram 241
11.2. Metric Service identification activity as BPMN diagram 242

13.1. BPMN diagram of the main process steps used to develop the metric system at the IT service provider 262
13.2. Prototypes for metric-based risk monitors: Risk Matrix and Open Risk Control 263
13.3. Static architecture overview of the data adapter and metric kernel of the EMI for risk metrics at our cooperation partner. 266
13.4. UML activity diagram for the data adaption for risk metrics. 267
13.5. UML activity diagram for the data storage and pre-calculation of the risk metric kernel. 268
13.6. UML class diagram for the data model of the risk metric kernel. 268
13.7. Prototype for the specific SSE Lab metric-based monitoring dashboard. . 272
13.8. Static architecture overview of the EMI core for the software project metrics in sse lab. 274
13.9. Screenshot of the realization of the SSE Lab metric-based monitoring dashboard 275
13.10Ticket flow visualized as Senkey diagram in the RiVER analysis tool (taken from [Cha12] p.75). 278
13.11Indication of the smell “loosing the battle of the inbox” using a radiogram visualization for the overview page of the RiVER analysis tool (taken from [Gji13] p.76). 279
13.12Specific ticket status changes visualized as work-item-history in the detail view of the RiVER analysis tool (taken from [Gji13] p.77). 280
13.13Static architecture overview of the first version of the RiVER analysis tool. 281
13.14Senkey diagram from the second version of the RiVER analysis tool showing approximately 1.5 million ticket status changes (taken from [Rab15] p.26). We drastically simplified the visualization (e.g. removed the names of the nodes) to make it more compact. 283
13.15Static architecture overview of the second version of the RiVER analysis tool. 284
13.16Comparison of the Senkey diagram provision speed of the two versions of the RiVER analysis tool based on the number of status changes in the databases (smaller is better). 285

14.1. Overview of the modeling workflow of the tool. 290
14.2. Screenshot of the metric documentation model in the documentation tool. 291
14.3. Screenshot of a web page of the generated metric documentation. 292
14.4. Screenshot of the overview page for a project in the metric management support tool. 293

14.5. Screenshot of the edit page for a monitor and its metrics in the metric management support tool. 294
14.6. Screenshot of the dashboard frontend from MeDIC Dashboard. 296
14.7. Screenshot of the dashboard frontend from SCREEN. 297
14.8. High level architecture overview of MeDIC Dashboard. 298
14.9. Architecture overview of SCREEN and its connection to the Render-Service and the Window-Service. 299
14.10Screenshot of the EMI Monitoring Service graphical user interface. 301
14.11Screenshot of the technical log view in the GUI of the EMI Logging Service (taken from [Dör14] p. 71) 302
14.12Screenshot of the logger configuration in the GUI of the EMI Logging Service (adapted from [Dör14] p. 68). 302
14.13Screenshot of the EMI Directory Service graphical user interface. 303
14.14Screenshot of the configuration of a specific renderer in the GUI of the EMI Render Service (adapted from [Röl13] p. 61). 304
14.15UML package diagram of parts of the common core of the EMI Framework.306
14.16Example for the implementation of a measurement cache using EMI Framework components. 307