

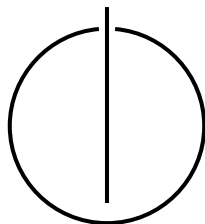


FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Wirtschaftsinformatik

**A GENERIC AND INTEGRATED SOFTWARE
SUPPORT FOR THE VISUALIZATION OF
METRICS IN THE CONTEXT OF
ENTERPRISE ARCHITECTURE
MANAGEMENT**

Michael Benjamin Walter Schätzlein





FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Wirtschaftsinformatik

**A GENERIC AND INTEGRATED SOFTWARE SUPPORT FOR
THE VISUALIZATION OF METRICS IN THE CONTEXT OF
ENTERPRISE ARCHITECTURE MANAGEMENT**

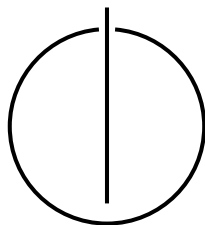
**GENERISCHE UND INTEGRIERTE SOFTWAREUNTERSTÜTZUNG FÜR
DIE VISUALISIERUNG VON METRIKEN IM KONTEXT VON
UNTERNEHMENSARCHITEKTURMANAGEMENT**

Author: Michael Benjamin Walter Schätzlein

Supervisor: Prof. Dr. rer. nat. Florian Matthes

Advisor: Dipl. Inform.-Univ. Ivan Monahov

Date: May 15th, 2014



I assure the single handed composition of this master's thesis only supported by declared resources.

München, May 15th, 2014

Michael Schätzlein

Abstract

These days, enterprise architects increasingly need quantitative enterprise architecture (EA) models as additional information foundation for decisions besides established qualitative models (e.g. EA domain models). Additionally, a growing number of stakeholders (e.g. compliance or risk management) need specific enterprise architecture management (EAM) metrics. In recent years an increasing number of publications were made on the subject of EAM metrics and performance indicators, however the aspect of metric visualization is not sufficiently addressed.

To close this gap, we develop a foundation for EAM metric visualizations in the conceptual part of this thesis. For this purpose, we analyze existing literature in different research areas (e.g. IT controlling, finance controlling) and suggest specific visualization types for metrics in the EAM domain. Furthermore, we discuss various strengths and weaknesses of the presented visualization types and additionally make suggestions on how specific EAM metrics should be visualized.

In the implementation part of the thesis we show a prototypical implementation of the concepts in an EAM tool developed by our group. This tool supports the definition and life-cycle management of metrics, but currently not the visualization. In addition to that we present the architecture of the implementation together with selected aspects (e.g. a template for the configuration of visualizations) and justify our architectural decisions.

To verify the usability of the solution, we implemented an instance in our tool for a collection of 52 metrics published by our group, and generated the visualizations on the base of test data.

Zusammenfassung

Heutzutage benötigen Unternehmensarchitekten verstärkt quantitative Unternehmensarchitektur(UA)-Modelle, neben bewährten qualitativen Modellen (z.B. Domänenmodellen), als weitere Informationsbasis für Entscheidungen. Zusätzlich benötigt eine immer größer werdende Zahl von Stakeholdern (z.B. Compliance oder Risk-Management) konkrete UA-Management-Kennzahlen. In den letzten Jahren wurde auch verstärkt zum Thema UAM-Kennzahlen und Performance Indicators publiziert, dennoch ist der Aspekt der Kennzahlen-Visualisierung nicht ausreichend adressiert.

Um diese Lücke zu schließen, erarbeiten wir im konzeptionellen Teil dieser Arbeit die Grundlagen für UAM-Kennzahl-Visualisierungen. Hierfür untersuchen wir zunächst die existierende Literatur in unterschiedlichen Forschungsgebieten (z.B. IT-Controlling, Finanzcontrolling) und schlagen konkrete Visualisierungstypen für Metriken in der UA-Domäne vor. Des Weiteren diskutieren wir unterschiedliche Stärken und Schwächen der vorgestellten Visualisierungstypen und geben zusätzlich Empfehlungen wie konkrete UAM-Kennzahlen dargestellt werden sollten.

Im Implementierungsteil der Arbeit zeigen wir eine prototypische Umsetzung dieser Konzepte in einem von unserer Gruppe entwickelten UAM-Werkzeug. Dieses Werkzeug unterstützt die Definition und das Lebenszyklus-Management, aber aktuell nicht die Visualisierung von Metriken. Desweiteren präsentieren wir die Architektur der Implementierung sowie weitere ausgewählte Implementierungsaspekte (z.B. ein Template zur Konfiguration von Visualisierungen) und begründen unsere Architekturentscheidungen.

Um die Nutzbarkeit der Lösung nachzuweisen haben wir für eine Sammlung von 52 Kennzahlen, publiziert von unserer Gruppe, jeweils eine Implementierung in unserem Werkzeug umgesetzt und auf Basis von Testdaten die Visualisierungen generiert.

Contents

- I. Introduction 1**
- 1. Motivation 3**
 - 1.1. Research questions 4
 - 1.2. Research method 5
 - 1.3. Outline 6
- 2. Introduction 9**
 - 2.1. Enterprise Architecture Management 9
 - 2.2. Metrics 11
 - 2.3. Metric visualizations 12
 - 2.4. Approach 13
- 3. Literature review 15**
 - 3.1. Identification of relevant literature 15
 - 3.2. Literature discussion 21
 - 3.3. Metrics of the Metric Catalog [Matthes et al., 2012a] 27
 - 3.4. Deriving requirements for the solution 30
- II. Conceptual part 33**
- 4. Recommended visualizations for the EAM domain 35**
 - 4.1. Line chart 36
 - 4.2. Column chart 37
 - 4.3. Pie chart 41
 - 4.4. Kiviat chart 42

4.5. Bullet chart	44
4.6. Critical reflection of the proposed visualization types	45
5. Prototypical implementation	49
5.1. Tricia	49
5.2. MxL 2.0	54
5.3. Identification & comparison of existing visualization engines	55
5.4. Iterative prototype development	60
6. Selected implementation aspects	69
6.1. Visualization template	69
6.2. MxL expressions	75
6.3. Configuration process	78
6.4. Rendering process	79
 III. Evaluation & Outlook	 83
7. Evaluation	85
7.1. Research questions	85
7.2. Requirements of the prototype	88
7.3. Critical reflection	89
8. Future research	93
8.1. Visualization of historical data	93
8.2. Dynamic configuration dialog	93
8.3. Guided metric selection	95
8.4. Graph visualizations	95
8.5. Graphical drill-down on visualizations	96
 Bibliography	 99
 A. Appendix	 i

List of Figures

1.1. Outline of the thesis.	7
2.1. Effects of EAM on the organization [Ahlemann et al., 2012].	10
2.2. DSR Knowledge Contribution Framework according to Gregor and Hevner [2013].	14
3.1. The three stages of effective literature review process according to Levy and Ellis [2006].	15
4.1. Example for a line chart.	36
4.2. Example for an area chart.	37
4.3. Example for a column chart.	38
4.4. Example for a bar chart.	39
4.5. Line chart of Figure 4.1 as column chart.	40
4.6. Exmple for a pie chart.	41
4.7. Pie chart of Figure 4.6 as column chart.	42
4.8. Example for a kiviatic chart.	43
4.9. Kiviatic chart of Figure 4.8 as column chart.	44
4.10. Example for a single metric visualized as gauge.	45
4.11. Example for a bullet chart according to Few [2006].	45
4.12. Example for a traffic light.	48
4.13. Example for a colored bullet chart.	48
5.1. Basic model hierarchy of Tricia according to Reschenhofer [2013].	50
5.2. Basic processing of a HTTP request to Tricia according to Reschenhofer [2013].	52
5.3. Exemplary <i>TypeDefinition</i> “ <i>Business Application</i> ” consisting of several <i>PropertyDefinitions</i>	53

5.4.	Abstract hybrid wiki data model according to Reschenhofer [2013].	53
5.5.	Class diagram of the configurator backend. (Private attributes omitted) . .	66
5.6.	Class diagram of the visualization configurator. (Methods omitted)	67
6.1.	Example visualization with annotations.	70
6.2.	Result of bullet chart configuration of Listing 6.5	74
6.3.	Result of bullet chart configuration of Listing 6.6	76
6.4.	Activity diagram for a new configuration dialog.	78
6.5.	Configuration dialog.	81
6.6.	Activity diagram for rendering a visualization.	82
8.1.	Resulting dialog fields of Listing 8.1.	95
A.1.	Full class diagram of the visualization configurator.	iii

Listings

5.1. Example for a <i>BlockSubstitution</i>	54
5.2. The metric <i>Application continuity plan availability</i> defined as MxL expression [Reschenhofer, 2013].	55
6.1. Empty visualization template	71
6.2. Line chart properties.	72
6.3. Bullet chart properties.	73
6.4. Empty configuration.	74
6.5. Embedded bullet chart configuration.	75
6.6. Embedded bullet chart configuration with “expert” customization.	76
6.7. Input data for the line chart in Figure 6.1.	77
6.8. Wrapper expression for <i>Application continuity plan availability</i>	77
8.1. Exemplary template addition for a dynamic configuration dialog.	94
A.1. Complete line chart template.	i

List of Tables

3.1. Identified literature	19
3.2. Literature classification by area of visualization.	20
3.3. Literature classification by subject.	21
3.4. Metrics of the Metric Catalog [Matthes et al., 2012a]	29
4.1. Properties of line charts.	37
4.2. Properties of column charts.	39
4.3. Properties of pie charts.	42
4.4. Properties of kiviatic charts.	43
4.5. Properties of bullet charts.	46
4.6. Recommended use of visualization types for the result types of metrics. . .	46
4.7. Visualization types as found in literature.	47
5.1. Evaluation of RGraph	56
5.2. Evaluation of g.Raphaël	57
5.3. Evaluation of NVD3	58
5.4. Evaluation of Highcharts	59
5.5. Visualization engine evaluation results.	60

Part I.

Introduction

1. Motivation

Metrics are an important instrument for managing organizations [Parmenter, 2010]. They allow to quantify economic aspects (e.g. Return on Investment [Farris et al., 2010]), in order to understand changes in the organization over time. Additionally, metrics support the planning of necessary changes to reach the organization's goals.

Especially in the EAM domain, metrics are more and more important [Matthes et al., 2012b,c; Monahov et al., 2013; Reschenhofer, 2013]. Therefore our group developed concepts and a software prototype (Tricia) to allow the development and calculation of EAM-specific metrics [Reschenhofer, 2013].

Problem statement

In the field of metric-based management, metric visualizations play an important role [Hanschke, 2010; Kerzner, 2013; Parmenter, 2010; Rawolle et al., 2008; Stutz, 2009; Wildemann, 1996]. Thus, typical visualization types for metrics are described in business administration and IT controlling literature. Nevertheless, as shown by our literature review (Chapter 3), hardly any information on this subject could be found for the EAM domain. Additionally, our tool currently does not support the graphical representation of user-defined EAM metrics and therefore this important aspect is not addressed.

Solution

To solve this problem, we suggest specific visualization types for metrics, based on recommendations of the relevant literature in areas related to the EAM domain. In addition to that, we examine several web-based libraries to support the web-based visualization

of user-defined metrics. Consequently, we present a prototypical extension of Tricia to support the suggested visualization types.

Contributions to the body of knowledge

Our solution provides three primary contributions to the body of knowledge in the field of EAM metrics:

1. A list of recommended visualization types for the graphical representation of EAM metrics (Chapter 4).

We thereby provide a recommendation based on the metrics of the Metric Catalog (Matthes et al. [2012a]), either for the representation of the current value of a metric, or for the representation of the metric values over time.

2. A comparison of web-based visualization libraries supporting the recommended visualization types (Section 5.3).

We compare four visualization libraries based on the recommended visualization types and on additional functionality.

3. A prototypical extension to Tricia supporting the visualization of user-defined metrics (Chapter 6).

The prototypical extension ties into previous work by Reschenhofer [2013] and offers full support for the recommended visualization types. Selected aspects are subject in Chapter 6.

1.1. Research questions

The research goal for this thesis is the development of an integrated software support for the user-defined metric visualization in the domain of EAM in Tricia. To reach this goal we define the following research questions:

Research question 1: What are recommended visualizations for metrics in literature?

Thereby, analyzing existing literature in the fields of management controlling, IT controlling, and project management, we aim to identify relevant visualization types for the visualization of metrics.

Research question 2: What type of visualizations are recommended for metrics in the domain of EAM?

Considering the results of the previous research, we aim to identify relevant visualization types specific to metrics in the domain of EAM.

Research question 3: What are suitable web-based visualization libraries to support the visualization of metrics in the domain of EAM?

As the solution based on Tricia has to provide web-based visualizations, we aim to examine different web-based visualization libraries to find a suitable candidate for our prototypical extension of Tricia.

Research question 4: What is a suitable prototypical implementation of a corresponding visualization extension of Tricia?

Finally, since our main goal is a prototypical extension of Tricia to support the visualization of user-defined metrics we aim to present relevant aspects of this prototypical implementation.

1.2. Research method

This thesis is based on three major research methods as guides for the structure and approach of the thesis and its parts:

- Design science research process proposed by Gregor and Hevner [2013] for the overall structure and approach of this thesis. (Subject in Section 2.4)

- Literature research process proposed by Levy and Ellis [2006] for the literature review. (Subject in Chapter 3)
- Evolutionary prototyping proposed by Davis [1992] for the prototypical implementation. (Subject in Section 5.4)

1.3. Outline

Figure 1.1 presents the outline of the thesis and its division into chapters and sections.

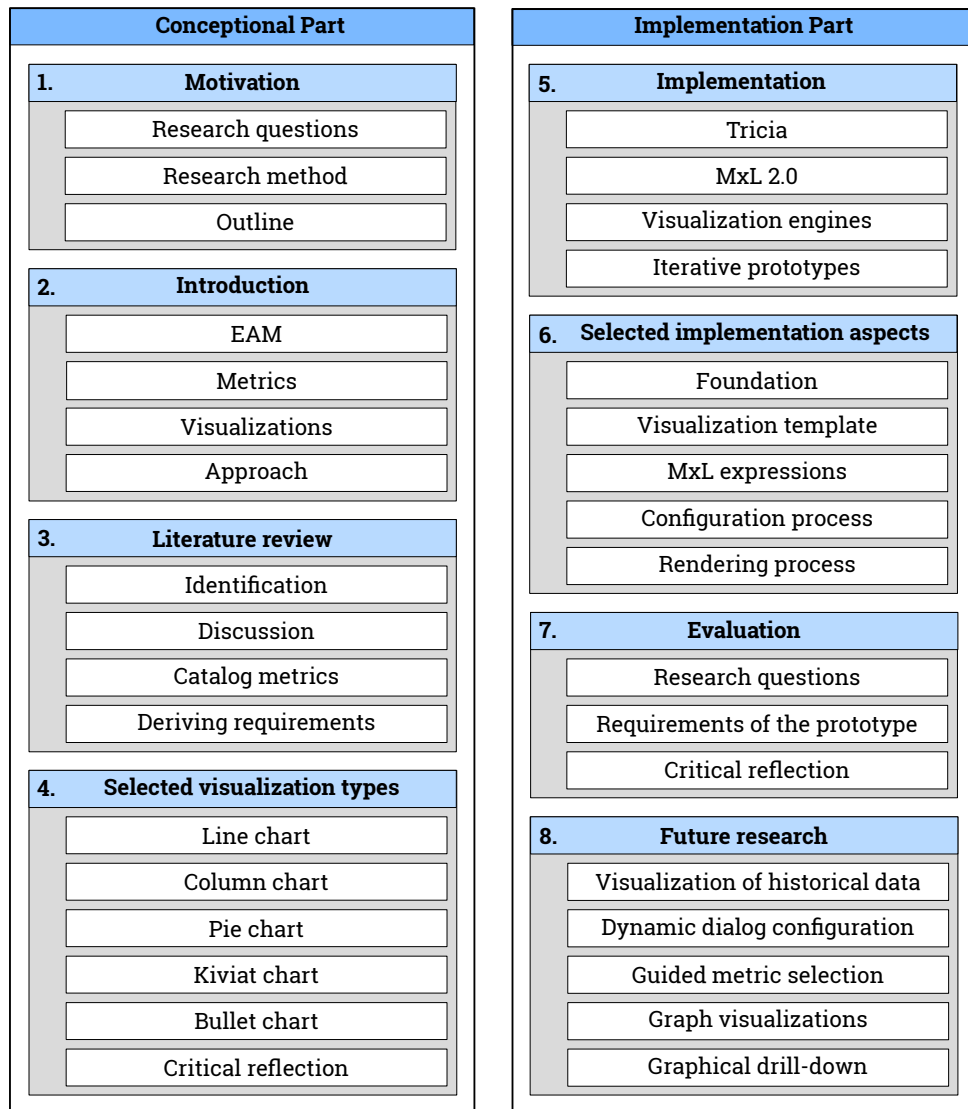


Figure 1.1.: Outline of the thesis.

2. Introduction

2.1. Enterprise Architecture Management

According to Buckl et al. [2010] an *Enterprise Architecture* (EA) can be defined as:

“The fundamental conception of the enterprise in its environment, embodied in its elements, their relationships to each other and to its environment, and the principles guiding its design and evolution.”

That is a holistic view of the organization, encompassing all areas from a business perspective to information technology (IT) aspects. In terms of an EA, holism means that observing only single elements of the organization and their relationships (e.g. only parts of the business perspective) is insufficient to understand the workings of the whole organization.

Organizations work in a continuously changing environment (e.g. variable customer demands, fast-paced technology innovation, shortening of product life cycles), therefore they have to constantly align their EA with these new conditions to stay ahead of competition [Ahlemann et al., 2012]. But if those changes do not ensue in a coordinated manner, they might introduce uncontrollable architectural complexity leading to unwanted side effects like for example loss of transparency, increased risks, and distraction from core business problems. This might lead to higher overall costs and decreased flexibility [Ahlemann et al., 2012].

Based on this ground, *Enterprise Architecture Management* (EAM) can be defined as:

“a management practice that establishes, maintains and uses a coherent set of guidelines, architecture principles and governance regimes that provide direction for and practical help with the design and the development of an enterprise’s architecture in order to achieve its vision and strategy.”

In other words a holistic way to understand, plan, develop and control an EA's evolution to preserve its transparency, flexibility, and efficiency [Ahlemann et al., 2012]. Hence EAM can help to increase overall performance of the organization as displayed in Figure 2.1.

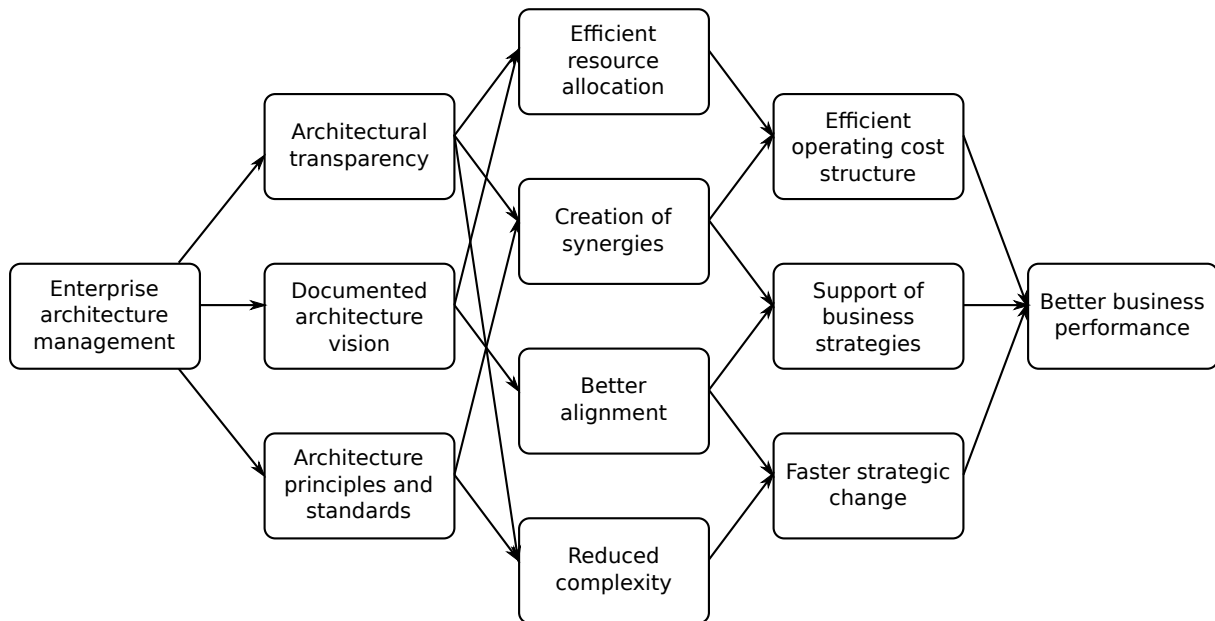


Figure 2.1.: Effects of EAM on the organization [Ahlemann et al., 2012].

Since the information need to support EAM and the complexity of the architecture is constantly growing, tool support for Enterprise Architecture Management becomes evermore important [Matthes et al., 2008]. These EAM tools help capturing, storing, structuring, analyzing, and communicating information belonging to the EA and its models [Short, 2013]. Usually the data pertaining to the EA exists in an unstructured form (e.g. spreadsheets, slides, notes) and are shared between many stakeholders [Buckl et al., 2009]. However the majority of existing EAM tools only offer very rigid information structures and collaboration mechanisms and this mismatch is a major problem [Matthes and Neubert, 2011].

To close this gap, our group developed the so called *Hybrid Wikis* [Neubert, 2012], which facilitate an incremental and collaborative enrichment of initially unstructured information with attributes, types, and integrity rules. The underlying enterprise wiki-system

called *Tricia*¹ subsequently allows architectural modeling, visualization, and analysis [Matthes and Neubert, 2011]. *Tricia* and its Hybrid Wikis are subject in Section 5.1.

2.2. Metrics

Due to the high number of model elements, their relationships, model element changes, and a high number of individual stakeholders, managing an Enterprise Architecture is a challenging task. In order to ease this task, it is necessary to measure, monitor, and evaluate certain performance-related characteristics of the EA [Lankes, 2008]. In literature, different terms are used to describe this concept [Subashi, 2013], for example: *Key Performance Indicator (KPI)*, *Performance Indicator (PI)*, *Indicator* and *Measurement*. However in this thesis the general term *Metric* will be used.

Although this metrics are an important means to assess the structure of the EA as well as the achievement of predefined EAM goals and thereby essential to develop meaningful value proposition [Lucke et al., 2010], there exists no common structure for the definition and documentation of those EAM metrics [Matthes et al., 2012c]. Furthermore, though well-known industry frameworks, e.g. ITIL [Office of Government Commerce (OGC), 2000] and CobiT [ITGI, 2009] provide an abundance of possible metrics, those metrics suffer from similar problems: The metrics either

- focus on a subset of EA management goals,
- are too generic,
- are documented and structured differently,
- do not detail on the required data,
- and/or cannot be adapted to the specific enterprise context [Matthes et al., 2012a].

Therefore our group developed an uniform and configurable *Metric Management Fact Sheet (MMFS)* consisting of general description elements as well as organization-specific description elements. Additionally our group published an *Metric Catalog* [Matthes et al., 2012a], an organized collection of 52 metrics observed in industry, uniformly described

¹<http://www.infoasset.de>

by the application of the MMFS. Moreover, Matthes et al. [2012b] developed a method for the holistic life-cycle management of metrics in the EAM domain accounting for both the MMFS and the Metric Catalog. Furthermore, the method is implemented within the EAM framework *BEAMS* developed by our group and described by Buckl [2011] and Schweda [2011].

Although the Metric Catalog provides a detailed and structured description for every metric, until recently a formal definition of the metric calculation was still missing. To close this gap, Monahov et al. created a domain-specific language (DSL) named *Model-based Expression Language* (MxL) [Monahov et al., 2013] which was later refined and enhanced in the master’s thesis of Thomas Reschenhofer [2013]. By the use of this DSL, all metrics of the catalog are definable in a formal way by stating the calculation specification as MxL expression and can be evaluated in an automated and tool-supported manner [Reschenhofer, 2013]. The current state of this DSL is subject in Section 5.2.

2.3. Metric visualizations

In general, information visualization is the “*use of interactive visual representations of abstract data to amplify cognition*” [Ware, 2012]. Here, the spatial representation is chosen as opposed to scientific visualization where the spatial representation is given [Munzner, 2008], whereby the use of graphs to present data is only a subsection of the information visualization discipline. The use of graphs to visualize data dates back to the 18th century, but for a long time it was a “*largely unscientific*” field [Cleveland and McGill, 1984].

Nevertheless, graphs are powerful tools for analyzing data and communicating quantitative information [Cleveland and McGill, 1985]. They provide the means to compare compacted data in similar fashion and to analyze a multitude of numbers in a short time frame [Botthof et al., 2008]. Additionally, visualizations allow the perception of emergent properties that were not previously anticipated [Ware, 2012].

Due to the controlling function of metrics in general (as well as in particular in the domain of EAM), proper graphical representations are required to support the user in the assessment of the current state of the EA and in guiding its evolution [Hanschke, 2010]. Therefore, the majority of available EAM tools support (metric) visualizations in different ways [Knoll and Schulz, 2013].

2.4. Approach

On the one hand there is a need for tools allowing the dynamic, formal, and structured definition of EAM metrics and their calculation, and on the other hand there is a need for tools supporting the proper visualization of said metrics. Unfortunately current tools offer support for either one of these requirements (e.g. Tricia), but not both [Knoll and Schulz, 2013]. Therefore, the goal of this thesis is the development of a generic and integrated software support for the visualization of metrics in the context of Enterprise Architecture Management as a *Design Science Research (DSR)* artifact.

In the field of *Information Systems (IS)*, DSR involves the construction of socio-technical artifacts such as decision support systems, modeling tools, and governance strategies as knowledge contribution [Gregor and Hevner, 2013]. The corresponding research method as proposed by Gregor and Hevner [2013] hereby consists of six steps, which are paralleled by the structure of the thesis:

1. Identify problem
2. Define solution objectives
3. Design and development
4. Demonstration
5. Evaluation
6. Communication

Each step has corresponding chapters and sections in the thesis. According to the DSR Knowledge Contribution Framework (illustrated in Figure 2.2) a knowledge contribution in DSR is possible in three different ways: improvement, invention, and exaptation. That is the development of new solutions for known problems, the invention of new solutions for new problems or the extension of known solutions to new problems. According to Gregor and Hevner [2013], applying known solutions to known problems is usually no knowledge contribution, even if done professionally.

The idea to use graphical representations to display metrics is known for decades [Antoine, 1956; Wildemann, 1996], nevertheless, the use of a formal and structured language for the

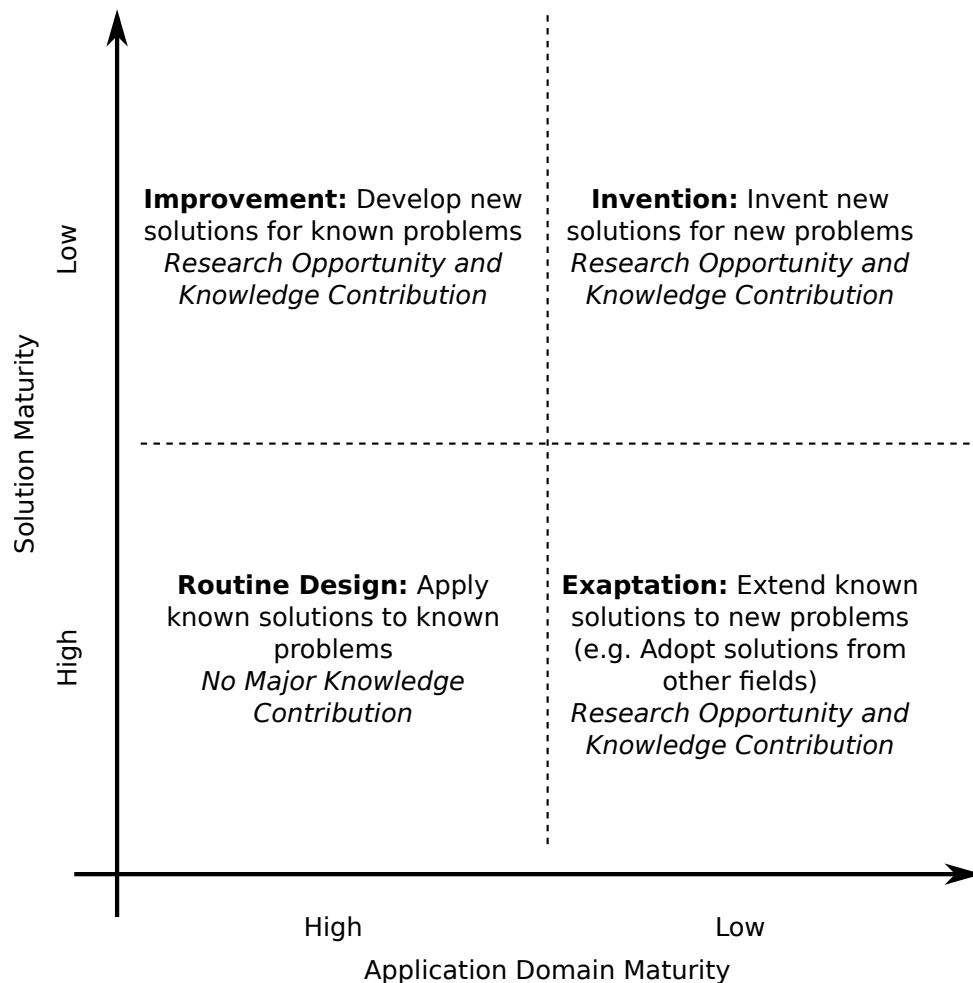


Figure 2.2.: DSR Knowledge Contribution Framework according to Gregor and Hevner [2013].

definition of metrics in the field of EAM is a quiet recent development [Monahov et al., 2013; Reschenhofer, 2013]. The thesis' artifact will provide a new solution (visualization of metrics in the field of EAM based on MxL) for a known problem (metric visualization) and can therefore be classified as *improvement* as defined by the DSR Knowledge Contribution Framework. It offers hereby a knowledge contribution and is a valid research project.

3. Literature review

3.1. Identification of relevant literature

As described in Section 2.4, the thesis is a research project of the IS discipline. The authors Levy and Ellis [2006] propose a three-step literature review process to guide an effective literature review in the field of IS. The process is divided into three steps as outlined in Figure 3.1:

1. Inputs (literature gathering and filtering)
2. Processing
3. Outputs (writing the literature review)

This systematic approach is used in the thesis to identify and analyze relevant literature. Therefore the process is described in detail in the following sections.

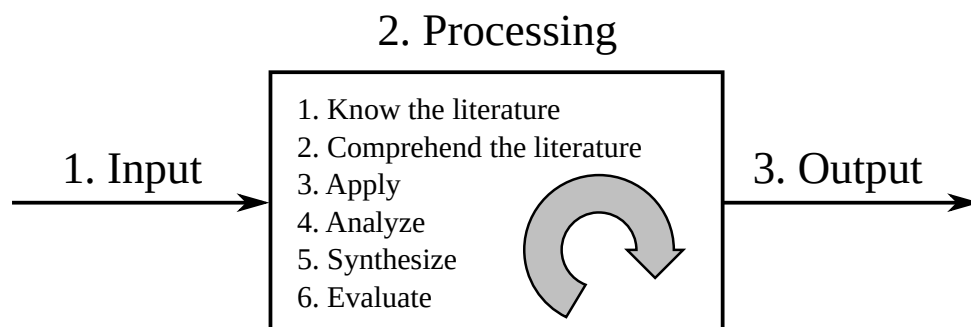


Figure 3.1.: The three stages of effective literature review process according to Levy and Ellis [2006].

Inputs

The focus of the literature review is to find information regarding the visualization of EAM metrics. Therefore contributions in the leading journals and in the electronic resources for library services were searched, as suggested by Levy and Ellis [2006]. Four different electronic libraries were hereby used: Google scholar², the IEEE digital library³, the ACM digital library⁴, and the digital library of CiteSeerX⁵. The research was conducted between November 2013 and February 2014 using the following three techniques suggested by Levy and Ellis [2006]:

1. Keyword search

The literature search was performed using the following search terms: *metric, KPI, PI, performance indicator, indicator, measurement, measure; visualization, graphical representation, representation, presentation, communication*, as well as possible combinations thereof in English as well as in German. The German *Kennzahl* was hereby used as translation for *performance indicator*.

2. Backward search

The backward search was performed in three sub-steps: *backward reference search* (review of the references of the articles resulting from the keyword search), *backward authors search* (review of prior publications of the authors resulting from the keyword search) and *previously used keywords* (review of the keywords resulting from the keyword search).

3. Forward search

The forward search was performed in two sub-steps: *forward references search* (review of additional article that have cited the article resulting from the keyword search), and *forward authors search* (review of articles the authors have published following the article resulting from the keyword search).

²<http://scholar.google.de>

³<http://ieeexplore.ieee.org>

⁴<http://dl.acm.org>

⁵<http://citeseerx.ist.psu.edu>

Processing

Subsequent to collecting sufficient literature, the sources have to be analyzed and screened. Hereby the publications were analyzed to determine the relevance of the literature to the subject of the thesis, irrelevant literature was dropped at this point. The resulting list is shown in Table 3.1.

Key	Title
[Ahlemann et al., 2012]	<i>Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments.</i> Springer, 2012.
[Botthof et al., 2008]	<i>Wie Zahlen wirken: Betriebliche Kennzahlen vorteilhaft darstellen.</i> Haufe-Lexware, 2008.
[Burger, 2006]	<i>Management Cockpit: Unterstützung von Kennzahlen durch eine effiziente Visualisierung.</i> LFI, Abt. für Handwerkswissenschaften, 2006.
[Cardoso, 2013]	<i>Challenges in performance analysis in enterprise architectures.</i> In Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International, pages 327–336. IEEE, 2013.
[Cleveland and McGill, 1984]	<i>Graphical perception: Theory, experimentation, and application to the development of graphical methods.</i> Journal of the American Statistical Association, 79(387):531–554, 1984.
[Cleveland, 1985]	<i>The elements of graphing data.</i> Wadsworth advanced books and software Monterey, CA, 1985.
[Cleveland and McGill, 1985]	<i>Graphical perception and graphical methods for analyzing scientific data.</i> Science, 229(4716):828–833, 1985.
[Few, 2006]	<i>Information dashboard design.</i> O’Reilly, 2006.
[Few, 2009]	<i>Now You See It: Simple Visualization Techniques for Quantitative Analysis.</i> Analytics Press, USA, 1st edition, 2009.
[Franceschini et al., 2007]	<i>Management by Measurement: Designing Key Indicators And Performance Measurement Systems.</i> Springer, 2007.

Key	Title
[Hanschke, 2009]	<i>Strategic IT Management: A Toolkit for Enterprise Architecture Management.</i> Springer, 2009.
[Hanschke, 2010]	<i>Strategisches Management der IT-Landschaft.</i> Hanser, München, 2010.
[Hanschke et al., 2013]	<i>Business-Analyse einfach und effektiv: Geschäftsanforderungen verstehen und in IT-Lösungen umsetzen.</i> Carl Hanser Verlag GmbH Co KG, 2013.
[Hanschke, 2013]	<i>Enterprise architecture management-einfach und effektiv: ein praktischer Leitfaden für die Einführung von EAM.</i> Carl Hanser Verlag GmbH Co KG, 2013.
[Kelley and Donnelly Jr, 2009]	<i>The Humongous Book of Statistics Problems: Translated for People Who Don't Speak Math.</i> Penguin, 2009.
[Kerzner, 2013]	<i>Project Management Metrics, Kpis, and Dashboards: A Guide to Measuring and Monitoring Project Performance.</i> John Wiley & Sons, 2013.
[Monahov et al., 2013]	<i>Design and prototypical implementation of a language empowering business users to define key performance indicators for enterprise architecture management.</i> In Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International, pages 337–346. IEEE, 2013.
[Parmenter, 2010]	<i>Key Performance Indicators: Developing, Implementing, and Using Winning KPIs.</i> John Wiley & Sons, 2010.
[Rasmussen et al., 2009]	<i>Business Dashboards: A Visual Catalog for Design and Deployment.</i> John Wiley & Sons, 2009.
[Rawolle et al., 2008]	<i>IT-basierte Prozesssteuerung – Geschäftsprozessmonitoring am Beispiel der Versicherungsindustrie.</i> Controlling & Management, 52:75–83, 2008.

3.1. Identification of relevant literature

Key	Title
[Reschenhofer, 2013]	<i>Design and prototypical implementation of a model-based structure for the definition and calculation of enterprise architecture key performance indicators.</i> Master's thesis, Technische Universität München, 2013.
[Stutz, 2009]	<i>Kennzahlen für Unternehmensarchitekturen: Entwicklung einer Methode zum Aufbau eines Kennzahlensystems für die wertorientierte Steuerung der Veränderung von Unternehmensarchitekturen.</i> PhD thesis, Universität St. Gallen, 2009.
[Subashi, 2013]	<i>Establishing kpi systems for enterprise architectures: Risks and countermeasures.</i> Master's thesis, Technische Universität München, 2013.
[Tiarks, 2008]	<i>Software-Monitoring mit Hilfe eines Dashboards.</i> Master's thesis, Universität Bremen, 2008.
[Tufté, 1991]	<i>Envisioning information.</i> Optometry & Vision Science, 68(4):322–324, 1991.
[Tufté, 2001]	<i>The Visual Display of Quantitative Information, volume 2.</i> Graphics Press Cheshire, CT, 2001.
[Wallin, 2009]	<i>Interactive balanced scorecard visualization.</i> Master's thesis, Chalmers University of Technology, 2009.
[Ware, 2012]	<i>Information Visualization: Perception for Design.</i> Elsevier, 2012.
[Wildemann, 1996]	<i>Visualisierung als Controlling-Instrument.</i> Neue Organisationsformen im Unternehmen. Ein Handbuch für das moderne Management, Berlin ua, pages 946–952, 1996.

Table 3.1.: Identified literature

Outputs

The sources identified in the previous step were classified into four distinct categories: literature describing the visualization of EAM metrics, literature describing the visualization of metrics, literature describing the general visualization of data, and literature in the field of EAM and metrics without mentioning visualizations. The result of this classification are shown in Table 3.2. It is clearly visible that no literature specific to the visualization of EAM metrics was unearthed during the research.

Criteria	Literature
Visualization of EAM metrics	-
Visualization of metrics	[Botthof et al., 2008], [Burger, 2006], [Few, 2006], [Kerzner, 2013], [Parmenter, 2010], [Rasmussen et al., 2009], [Tiarks, 2008], [Wallin, 2009]
General visualization of data	[Cleveland, 1985], [Cleveland and McGill, 1984, 1985], [Few, 2009], [Kelley and Donnelly Jr, 2009], [Tufte, 1991, 2001], [Ware, 2012]
Deems visualizations of metrics as necessary	[Hanschke, 2013], [Hanschke, 2010], [Monahov et al., 2013], [Rawolle et al., 2008], [Reschenhofer, 2013], [Stutz, 2009], [Subashi, 2013], [Wildemann, 1996]
No mention of visualization	[Ahlemann et al., 2012], [Cardoso, 2013], [Franceschini et al., 2007], [Hanschke et al., 2013], [Hanschke, 2009]

Table 3.2.: Literature classification by area of visualization.

Another way to classify the literature is to group it based on their subject. Table 3.3 shows such a classification.

Subject	Literature
Information Visualization	[Cleveland, 1985], [Cleveland and McGill, 1984, 1985], [Few, 2006], [Few, 2009], [Tufte, 1991, 2001], [Ware, 2012]
IT Controlling	[Ahlemann et al., 2012], [Cardoso, 2013], [Hanschke, 2009, 2010, 2013], [Monahov et al., 2013], [Reschenhofer, 2013], [Stutz, 2009], [Subashi, 2013], [Tiarks, 2008]
Management Controlling	[Botthof et al., 2008], [Burger, 2006], [Franceschini et al., 2007], [Hanschke et al., 2013], [Parmenter, 2010], [Rasmussen et al., 2009], [Rawolle et al., 2008], [Wallin, 2009], [Wildemann, 1996]
Statistics	[Kelley and Donnelly Jr, 2009]
Project Management	[Kerzner, 2013]

Table 3.3.: Literature classification by subject.

3.2. Literature discussion

The following section provides a description of the relevance of selected publications to the thesis and the implementation. The publications are hereby selected from the categories *Visualization of EAM metrics*, *Visualization of metrics*, and *General visualization of data*, as shown in Table 3.2.

Botthof et al. [2008] – Wie Zahlen wirken: Betriebliche Kennzahlen vorteilhaft darstellen

This book is an informal guide for the effective presentation of operational performance indicators. The main focus is the gathering of necessary information to conduct presentations on the state of the organization and hereby advises how to visualize appropriate metrics, too. The authors suggest the use of column charts and bar charts to illustrate

value changes over time, and the use of pie charts to illustrate value ratios. For the thesis, only the suggested visualization types are relevant for Chapter 4.

Burger [2006] – Management Cockpit: Unterstützung von Kennzahlen durch eine effiziente Visualisierung

This report (*Management Cockpit: Support of metrics with an efficient visualization*) is a guide directed at craft organizations about the visualization of appropriate metrics. It offers a few metrics with an in-depth description about their use and calculation directed at craft organizations. As visualization for the metrics, the author suggests kiviatic charts to illustrate value changes over time at different locations, pie charts to illustrate value ratios, and the use of column charts and line charts to illustrate the value change of a metric over time. Additionally he suggests the use of gauges to illustrate the current value of a metric (probably together with a target or historic value). For the thesis, only the suggested visualization types are relevant for Chapter 4.

Cleveland [1985] – The elements of graphing data

This book is a comprehensive guide to the visualization of data in general. It is a collection of best-practice recommendations and provides many in-depth descriptions of graphs and their usage. The content of the book is based on previous publications, e.g. [Cleveland and McGill, 1984] and [Cleveland and McGill, 1985]. Though the book is not specifically aimed at the visualization of metrics, the author generally suggests the use of line and column charts to illustrate value changes over time and advises against the use of pie charts and gauges. Cleveland suggests to use column charts instead. For the thesis, the suggested visualization types are relevant and Cleveland's arguments against pie charts and gauges provide valuable insights for the visualization selection in Chapter 4. In addition to that, the presented best-practices for visualization design have to be considered for the choice of an appropriate visualization engine in Section 5.3.

Few [2006, 2009] – Information dashboard design / Now you see it

Both books are detailed guides for the visualization of data and combinations thereof. Whereas *Now You See It* is about data visualization in general, *Information dashboard design* is specific to the combination of metric visualizations into meaningful dashboards for controlling purposes. The author suggests the use of line charts and column charts to illustrate the change of metric values over time and strongly advises against the use of pie charts and gauges. Instead of pie charts, column charts, and instead of gauges, bullet charts are to be used, the latter defined by the author himself. Additionally, both books contain advice regarding the design of visualizations, e.g. no distortion of data, no split axes, consistent layouts, etc. For the thesis, the suggested visualization types, especially the newly introduced bullet chart, are relevant and Few's arguments against pie charts, gauges, and kiviart charts provide valuable insights for the visualization selection in Chapter 4. Additionally, the presented design recommendations have to be considered for the choice of an appropriate visualization engine in Section 5.3

Kelley and Donnelly Jr [2009] – The humongous book of statistics problems

This book is a guide to different statistics problems with a subsection for descriptive statistics and thus different charts. The authors focus on the use of statistics-specific charts (e.g. histograms), but give some advice about the general visualization of data series. The authors suggest the use of pie charts to illustrate the ratio between values and line charts for the change of values over time. For the thesis, only the suggested visualization types are relevant.

Kerzner [2013] – Project Management Metrics, Kpis and Dashboards

This book is a comprehensive guide to dashboard design, going from the definition of metrics to the finished management dashboard. The author provides many examples for metric visualizations, but mostly as negative examples. Nevertheless he suggests the use of line charts and column charts to illustrate value changes over time and the use of gauges to illustrate the current value of a metric. Since this thesis does not focus on the

combination of visualizations to metric dashboards only the presented visualizations are relevant for Chapter 4.

Parmenter [2010] – Developing, Implementing, and Using winning KPIs

This book is a guide for the implementation of a metric system in an organization. Metric visualizations are mentioned several times throughout the book, mostly in the form of examples and a few general guidelines. The presented visualization types are line charts and column charts to illustrate value changes over time and gauges to illustrate the current and last two historic values of a metric. For the thesis, only the suggested visualization types are relevant for Chapter 4.

Rasmussen et al. [2009] – Business Dashboards

This book is a comprehensive guide for the implementation of business dashboards. The focus is on the the selection of appropriate metrics for a dashboard and the implementation of a dashboard (technical as well as organizational prerequisites). Metric visualizations are presented implicit by example. The presented visualization types are column charts to illustrate the value of a metric for different locations, line charts and column charts to illustrate value changes over time, and gauges to illustrate the current value of a metric. Additionally the authors present pie charts to represent the ratio between values and bubble charts without a use case. For the thesis, only the suggested visualization types are relevant for Chapter 4.

Tiarks [2008] – Software-Monitoring mit Hilfe eines Dashboards

This master's thesis (*Software-Monitoring with help of a dashboard*) develops a dashboard to visualize selected software metrics and validates the results with a case study. In the course of this, the author presents bubble charts, pie charts, line charts, column charts, histograms, and kivi charts, but the author does not describe how this charts can be employed to visualize metrics. For the thesis, only the suggested visualization types are relevant for Chapter 4.

Tufte [2001, 1991] – The Visual Display of Quantitative Information / Envisioning information

Both books [Tufte, 2001, 1991] are well-cited⁶ publications in the field of information visualization. Whereas *Envisioning information* is more about the display of complex data, *The Visual Display of Quantitative Information* is about graphs and their usage throughout history and therefore more relevant to the topic of the thesis. Both books are best-practice collections of existing visualizations with general guidelines and negative examples in between. Though both books are not specifically aimed at the visualization of metrics, the author generally suggests the use of line and column charts to illustrate value changes over time and advises against the use of pie charts and gauges. Similar to Cleveland [1985], the author suggests to use column charts instead. For the thesis, the suggested visualization types are relevant and again the arguments against pie charts and gauges provide valuable insights for the visualization selection in Chapter 4. In addition to that, similar to Cleveland [1985] and Few [2006, 2009], Tufte presents several best-practices for visualization design. Those recommendations have to be considered for the choice of an appropriate visualization engine in Section 5.3 as well.

Wallin [2009] – Interactive Balanced Scorecard Visualization

This master's thesis is a study with the goal to develop an interactive scorecard visualization. It has a strong focus on technology and implementation and uses a tree view for the scorecard visualization. Besides that, the author presents gauges to illustrate the current value of a metric in a couple of examples. For the thesis only the gauges are relevant for Chapter 4.

Ware [2012] – Information Visualization

This book [Ware, 2012] is an in-depth introduction into the field of information visualization research. It covers the theoretical foundations in great detail from the workings

⁶7657 respectively 4218 citations according to Google scholar.

of the human vision to the technical limitations. However the focus is not on the design of graphs as a visualization of data and therefore the author does not present any visualization applicable to the field of metric visualization. For the thesis, the book only contributes to the foundation of metric visualizations in general, but has no further use for the conceptual part.

3.2.1. Conclusion

Overall the literature review provides valuable insights into the state of the art of metric visualization. First of all, there seems to be no literature dedicated to the meaningful visualization of EAM metrics, as shown in Table 3.2. Admittedly, many researchers in the field of EAM use metrics to monitor the EA [Ahlemann et al., 2012; Cardoso, 2013] and propose visualizations as a tool to communicate the results [Hanschke, 2010, 2013; Monahov et al., 2013; Reschenhofer, 2013], however none of them states *how* to use visualizations to properly represent metrics. This is especially remarkable due to the fact that visualizations are routinely used to model the EA itself [Ahlemann et al., 2012; Hanschke, 2013; Wittenburg, 2007].

More in-depth guides to the visualization of metrics come from the field of business administration [Botthof et al., 2008; Burger, 2006; Parmenter, 2010; Wallin, 2009] and often focus on the design of controlling dashboards [Kerzner, 2013; Rasmussen et al., 2009; Tiarks, 2008], but those publications mostly lack a scientific backing of their statements. As a result, more scientific literature [Cleveland and McGill, 1984; Cleveland, 1985; Few, 2006, 2009; Tufte, 2001; Ware, 2012] tends to contradict those propositions. For example authors with a business administration or project management background tend to utilize gauges for the visualization of a single value [Kerzner, 2013; Parmenter, 2010], whereas authors with a scientific background tend to label them as a “*waste of space*” [Few, 2006] and “*hard to decode*” [Cleveland, 1985] and refrain from visualizing a single value altogether.

In summary, there is a gap between the people utilizing visualizations and those doing research on them and a research opportunity for visualizations in the field of EAM.

3.3. Metrics of the Metric Catalog [Matthes et al., 2012a]

The baseline for the supported visualizations will be the Metric Catalog, that is the artifact should offer proper visualization techniques for the representation of *at least* all metrics in the catalog. The catalog is hereby chosen, because it offers a large collection of metrics from different sources specific to the area of EAM. On top of that, the MMFS utilized in the catalog provides a calculation specification as well as additional information like e.g. a measurement frequency. The utility of this additional information will be shown later in this section.

As stated in Section 2.2, all metrics of the catalog can be defined as MxL expression, whereby the return value of the expression is the value of the corresponding metric. An overview of all 52 metrics defined in the catalog and the *type* of their calculation result is shown in Table 3.4.

Code	Name	Result Type
EAM-KPI-0001	Application continuity plan availability	percentage
EAM-KPI-0002	Backupped key roles	percentage
EAM-KPI-0003	Service portfolio methodology analysis	percentage
EAM-KPI-0004	Costs of inadequate change specifications	monetary value
EAM-KPI-0005	Business case quality	percentage
EAM-KPI-0006	Project's employee and contractor mix	unclear
EAM-KPI-0007	SLAs met	percentage
EAM-KPI-0008	Forecast quality	percentage
EAM-KPI-0009	Project performance index	multiple values
EAM-KPI-0010	PM guideline adherence	percentage
EAM-KPI-0011	SLA diffusion	percentage
EAM-KPI-0012	Application criticality ratings	percentage
EAM-KPI-0013	Incident duration	multiple values
EAM-KPI-0014	Customer satisfaction index	number
EAM-KPI-0015	IT process standard adherence (application)	percentage

Code	Name	Result Type
EAM-KPI-0016	Project compliance to target architecture	percentage
EAM-KPI-0017	Previously identified risks occurred	percentage
EAM-KPI-0018	Not previously identified risks occurred	percentage
EAM-KPI-0019	Workplace inspection	percentage
EAM-KPI-0020	IT processes measured by KPIs	percentage
EAM-KPI-0021	Project's quality plan availability	percentage
EAM-KPI-0022	Projects with quality manager \neq project manager	percentage
EAM-KPI-0023	Audit findings	number
EAM-KPI-0024	Employee in strategic focus areas	percentage
EAM-KPI-0025	Skill profile description availability	percentage
EAM-KPI-0026	Employee qualification	percentage
EAM-KPI-0027	Employee satisfaction index	number
EAM-KPI-0028	IT staff training	percentage
EAM-KPI-0029	Employees in innovative projects	percentage
EAM-KPI-0030	Business domain coverage of target architecture	percentage
EAM-KPI-0031	Application portfolio methodology analysis	percentage
EAM-KPI-0032	IT process standard adherence (service)	percentage
EAM-KPI-0033	Business application technology standards compliance	percentage
EAM-KPI-0034	Feasibility study performance index	percentage
EAM-KPI-0035	IT responsiveness satisfaction index	percentage
EAM-KPI-0036	IT investment delivering predefined benefits	percentage
EAM-KPI-0037	IT roles staffed	percentage
EAM-KPI-0038	Background checks	percentage

3.3. Metrics of the Metric Catalog [Matthes et al., 2012a]

Code	Name	Result Type
EAM-KPI-0039	Defects uncovered prior to production	percentage
EAM-KPI-0040	Action plan for critical IT risks	percentage
EAM-KPI-0041	Feasibility study satisfaction index	percentage
EAM-KPI-0042	Maintenance projects effort	percentage
EAM-KPI-0043	Business applications compliant with IT architecture and technology standards	percentage
EAM-KPI-0044	Procurement policies compliance	percentage
EAM-KPI-0045	Service desk calls caused by inadequate training	percentage
EAM-KPI-0046	IT continuity plans for business applications supporting critical processes	percentage
EAM-KPI-0047	Unexpected service interruption duration	number
EAM-KPI-0048	Password standard compliance	percentage
EAM-KPI-0049	Reopened incidents	percentage
EAM-KPI-0050	Critical IT processes monitoring	percentage
EAM-KPI-0051	KPI targets met	percentage
EAM-KPI-0052	IT component category standardization	percentage

Table 3.4.: Metrics of the Metric Catalog [Matthes et al., 2012a]

As can be seen, most metrics output a percentage (e.g. *Application continuity plan availability*), some output a monetary value (e.g. *Costs of inadequate change specifications*), some output a value without unit, (e.g. *Customer satisfaction index*), some output multiple values (e.g. *Incident duration*), and one metric isn't clearly specified (*Project's employee and contractor mix*).

In addition to the result type, most metrics have a recommended *measurement frequency*, that is how often the metric should be calculated. Consequential metrics therefore can be viewed as a snapshot, that is just the current value of the metric, over time, that is the

change of the metric over time, or in comparison of different organizations/different parts of an organization, e.g. the value of the metric per subdivision. For the representation of the metric value over time and in comparison, additional MxL expressions are needed to calculate the value of a metric at a specific point in time and for a single comparison unit respectively.

3.4. Deriving requirements for the solution

The following section defines the functional requirements for the implementation considering the findings of the literature review (Section 3.2), the metrics defined in the Metric Catalog (Section 3.3), and previous works, especially MxL (Section 2.2). Due to the fragmentary content regarding the visualization of metrics in literature, the requirements are complemented where necessary. First, all requirements are listed and subsequently explained in detail:

1. Support the visualization of all metrics from the Metric Catalog by all types of recommended visualizations.
2. The visualizations operate on the results from related MxL expressions.
3. Integration of the visualization prototype in Tricia.
4. The prototype must enable users to define visualizations for their metrics.

3.4.1. Support the visualization of all metrics from the Metric Catalog

As described in Section 3.3, the baseline for the implementation will be the metrics of the Metric Catalog, that is, at least those visualizations (e.g. line charts, column charts) that are needed to display the metrics of the catalog have to be supported. Since every metric can be displayed in various ways (snapshot, over time, etc.) and various formats (line chart, area chart, etc.), multiple, possibly equivalent, visualization types are necessary. An overview over all supported visualization types and the recommendations for the Metric Catalog can be found in Chapter 4.

3.4.2. The visualizations operate on the results from related MxL expressions

As a consequence of the previous requirement, the visualizations have to operate on MxL expressions for the definition and calculation of metrics. Section 2.4 states the need for a tool supporting both the structured and formal definition and calculation and the proper visualization of EAM metrics. MxL was specifically designed with the metrics of the Metric Catalog in mind it is therefore the first choice when it comes to the calculation and visualization of said metrics. Hence, the implementation has to be able to calculate MxL expressions and visualize the results. MxL is subject in Section 5.2.

3.4.3. Integration of the visualization prototype in Tricia

Tricia is an enterprise wiki-system developed by infoAsset and the sebis chair and offers a working implementation of MxL called *TxL* based on its flexible information model [Reschenhofer, 2013]. Therefore Tricia is a prime candidate as a foundation for the implementation of metric visualizations based on MxL. The prototype therefore has to be integrated into Tricia, which is subject in Section 5.1.

3.4.4. The prototype must enable users to define visualizations for their metrics

Due to different stakeholders having different needs and preferences [Botthof et al., 2008], the visualizations mustn't be predefined for every metric. Instead, the users must be able to define their own visualizations, considering their personal and organizational preferences. The prototype therefore has to provide the facilities to define individual visualizations on a per-user basis. That is, every user should be able to define an individual visualization for every metric.

Part II.

Conceptual part

4. Recommended visualizations for the EAM domain

Over the years, people have developed many different kinds of graphical representations for their data, depending on the structure of the data, the intended receiver, and the individual sense of aesthetics [Cleveland, 1985; Tufte, 2001]. In addition to that, there is no consensus on which visualizations are most suited to the task as stated in Section 3.2.1. The following chapter therefore does not try to include every possible design, but to present a small selection of appropriate visualizations found in literature. Their suitability is thereby measured against the metrics of the Metric Catalog as stated in Section 3.3. Every visualization is shortly described by the following properties:

Name:	The name of the visualization type. This name will be used to refer to the visualization throughout the following chapters and the implementation.
X-Axis:	Type of the x-axis. Possible characteristics are <i>category</i> for an axis with nominal values (e.g. names) or <i>date & time</i> for an interval scale with date and/or time values.
Y-Axis:	Type of the y-axis. Possible characteristics are <i>linear</i> for a linear scaled axis or <i>logarithmic</i> for a logarithmic (base 10) scaled axis.
No. of Metrics:	Whether only <i>one</i> or <i>multiple</i> metrics can be visualized in the same diagram.
Purpose:	Whether the visualization can be used to display a <i>snapshot</i> of one or multiple metrics or to display one (or multiple) <i>metric over time</i> .

4.1. Line chart

A line chart is a visualization which displays a series of data as points connected by straight lines. It hereby shows how the displayed value changes over time, as a consequence the x-axis has to consist of consecutive date values. Similar to line charts are area charts, with the sole difference being the area below the line, which is filled in with a color in area charts. For the visualization of metrics, line and area charts can be utilized to show trends and patterns in the value of a metric over time. As an example, Figure 4.1 shows an example visualization for the metric *Application continuity plan availability* from the Metric Catalog. The same metric with the same values is shown again in Figure 4.2 encoded as an area chart.

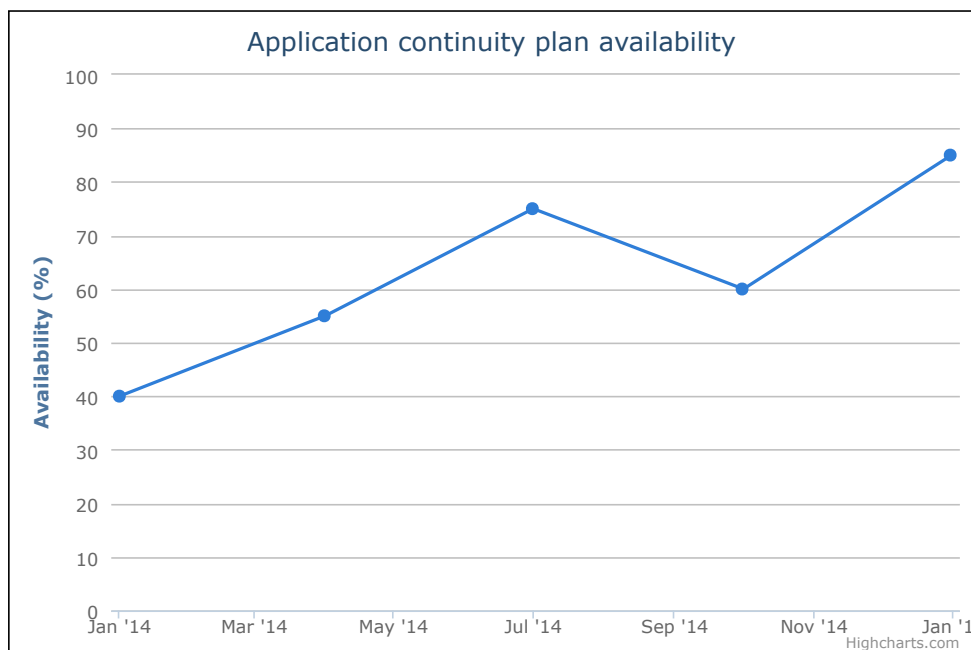


Figure 4.1.: Example for a line chart.

It is possible to display several metrics in the same visualization as long as they have the same result type. Otherwise more than one y-axis would be needed which might lead to confusion and show patterns where none exist [Tuft, 2001]. For metrics having a percentage as result type, the y-axis should be scaled linear from 0% to 100%, for monetary values (e.g. EAM-KPI-0004) or numbers (e.g. EAM-KPI-0014) the axis might

4.2. Column chart

be scaled logarithmic to emphasize percent change or multiplicative factors [Cleveland, 1985].

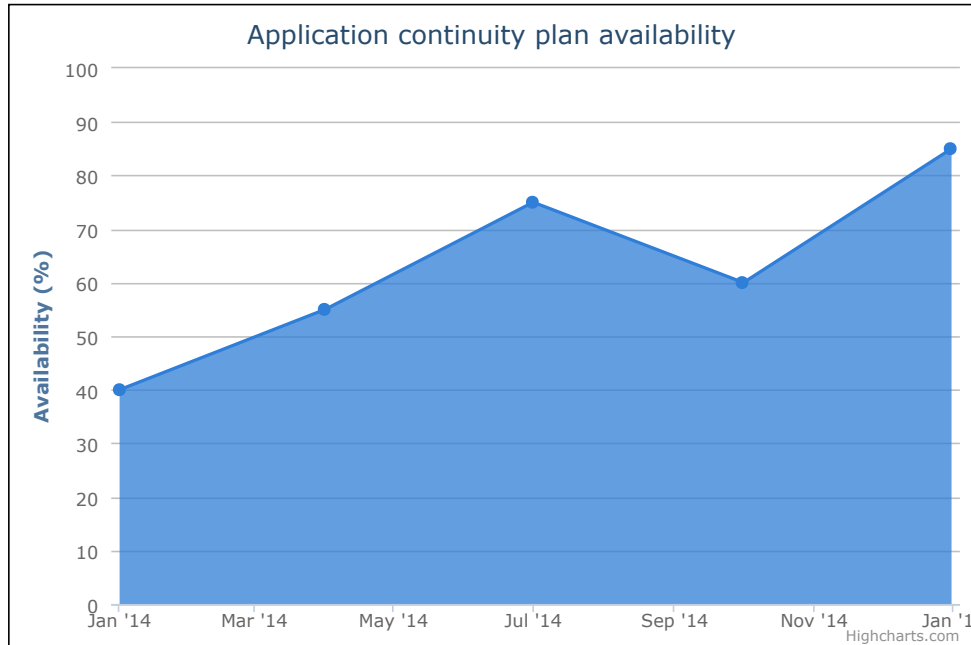


Figure 4.2.: Example for an area chart.

Name:	line/area chart
X-Axis:	date & time
Y-Axis:	linear or logarithmic
No. of Metrics:	multiple
Purpose:	metric over time

Table 4.1.: Properties of line charts.

4.2. Column chart

A column chart is a visualization which displays a series of data as rectangular bars with lengths proportional to the represented value. The bars can be plotted vertical or horizontal, for the remainder of the thesis, charts with vertical bars are called *column*

charts and charts with horizontal bars are called *bar charts*. Column charts can be utilized to either display a snapshot of a metric with multiple values as result type (e.g. EAM-KPI-0013) as shown in Figure 4.3 and as a bar chart in Figure 4.4 or similar to a line chart to display the value change of a metric over time. In the latter case, multiple metrics can be combined into one visualization for easier comparison as shown in Figure 4.5. In

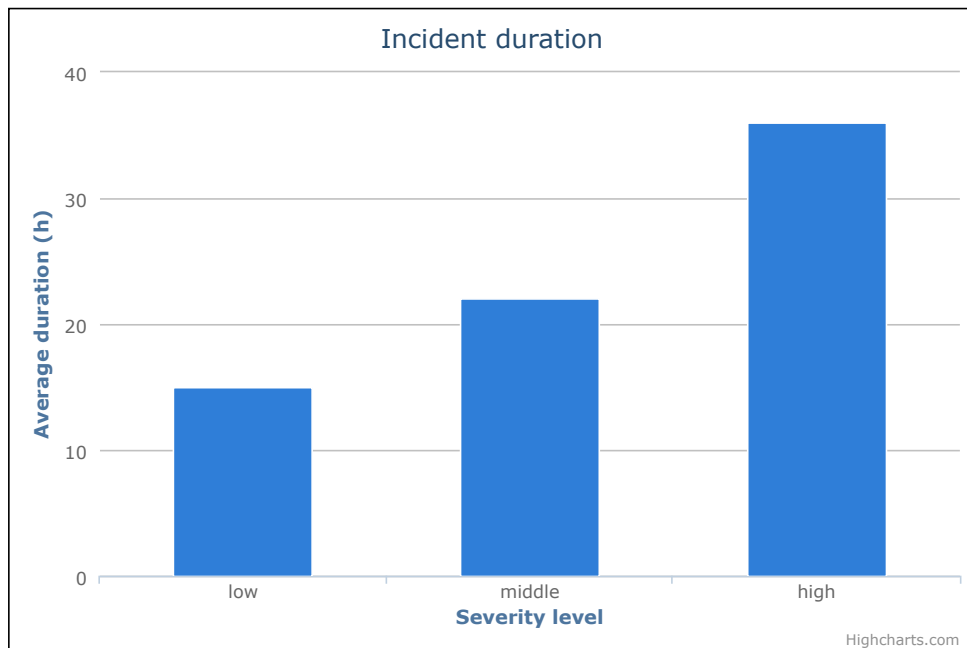


Figure 4.3.: Example for a column chart.

contrast to line charts it is important that the y-axis is scaled linear and never logarithmic, because otherwise the meaning of the graph will be distorted for the reader [Few, 2009]. For example in a log scale with a base of 10, a bar encoding a value of 100 will be half as long as a bar encoding a value of 10000, though it only represents $1/100^{th}$ of the value [Few, 2009]. Note that for bar charts the axis are swapped, i.e. bar charts are column charts rotated by 90 degree.

4.2. Column chart

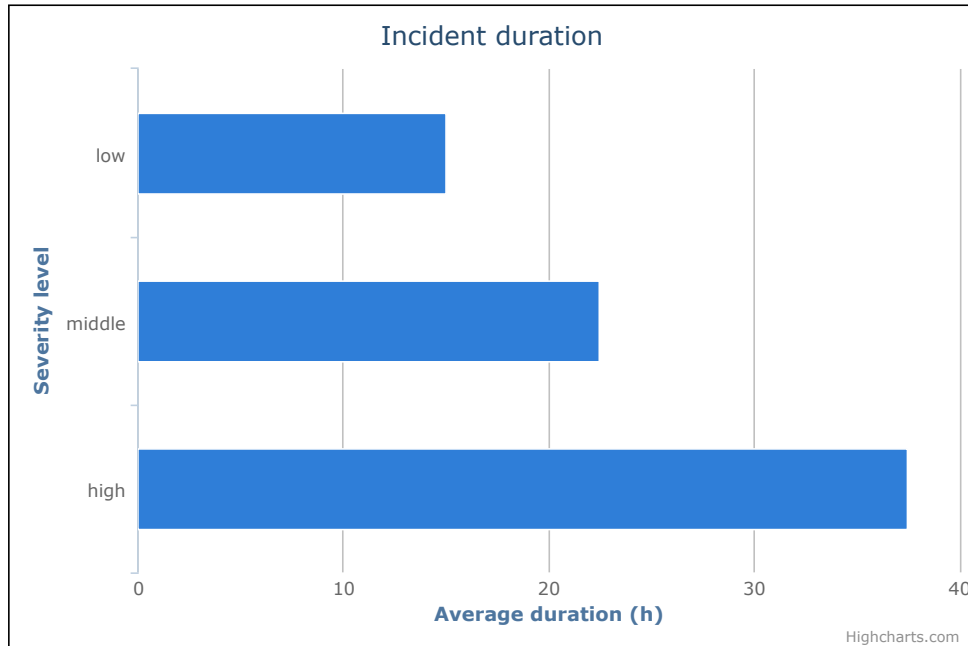


Figure 4.4.: Example for a bar chart.

Name:	column/bar chart
X-Axis:	categories or date & time
Y-Axis:	linear
No. of Metrics:	multiple
Purpose:	metric over time or snapshot

Table 4.2.: Properties of column charts.

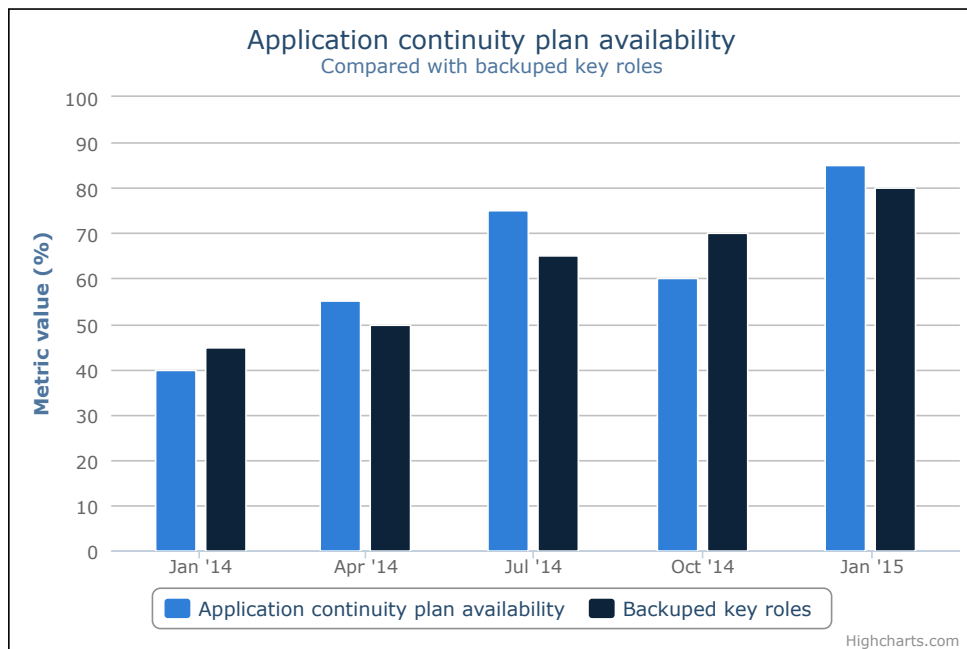


Figure 4.5.: Line chart of Figure 4.1 as column chart.

4.3. Pie chart

A pie chart is a circular chart divided into sectors, whereas the arc length of each sector is proportional to the value it represents. Usually pie charts are utilized to represent percentages, where the values of the slices sum up to 100 percent.

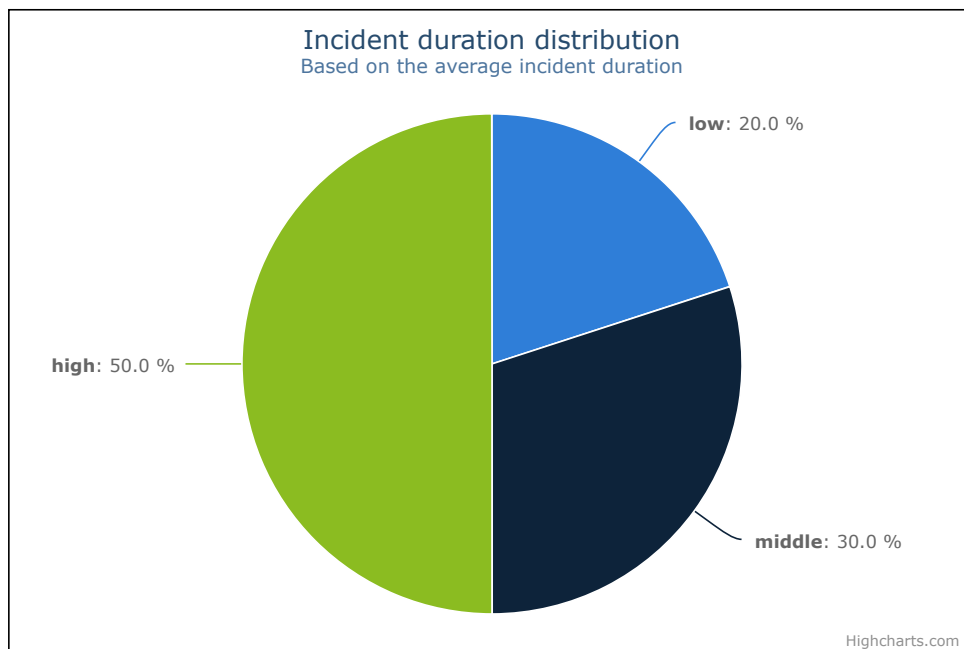


Figure 4.6.: Exmple for a pie chart.

Pie charts are widely used throughout the industry and the mass media [Cleveland, 1985], but many experts strongly refuse the use of pie charts due to the difficult decoding and comparison of values within and between charts [Cleveland, 1985; Few, 2006, 2007; Tufte, 2001]. In the words of Edward Tufte: “*The only worse design than a pie chart is several of them, for then the viewer is asked to compare quantities located in spatial disarray both within and between pies.*” [Tufte, 2001]. On top of that, the data displayed in a pie chart can often be better presented in a table [Few, 2007; Tufte, 2001] or in a column chart.

For the visualization of metrics, pie charts can be utilized to represent the proportion of values for metrics with multiple values as result type as shown in Figure 4.6, but as stated above, the same data can be represented with a bar chart as shown in Figure 4.7 without losing information.

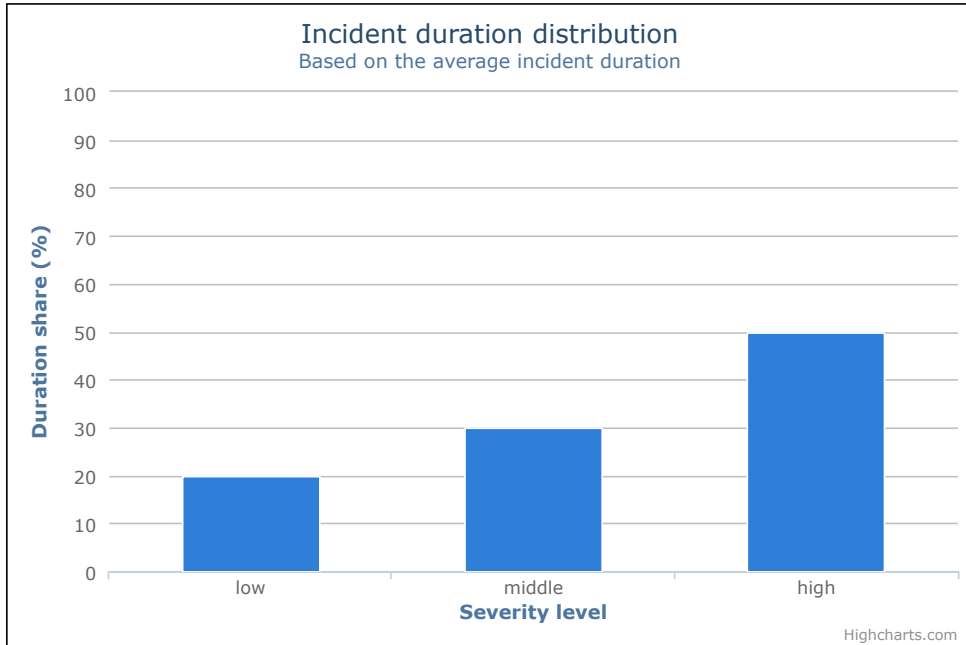


Figure 4.7.: Pie chart of Figure 4.6 as column chart.

Name:	pie chart
X-Axis:	categories
Y-Axis:	linear
No. of Metrics:	one
Purpose:	snapshot

Table 4.3.: Properties of pie charts.

4.4. Kiviatic chart

A kiviatic or spiderweb chart (also known as radar chart) is a radial visualization to display multivariate data. Usually it is utilized to compare objects on the basis of multiple criteria. Hereby every criteria is represented with a spoke and the value for each object is marked with a dot. All dots belonging to one object are connected with straight lines similar to a line chart. Basically, kiviatic charts are line charts transferred into a polar coordinate system.

4.4. Kiviat chart

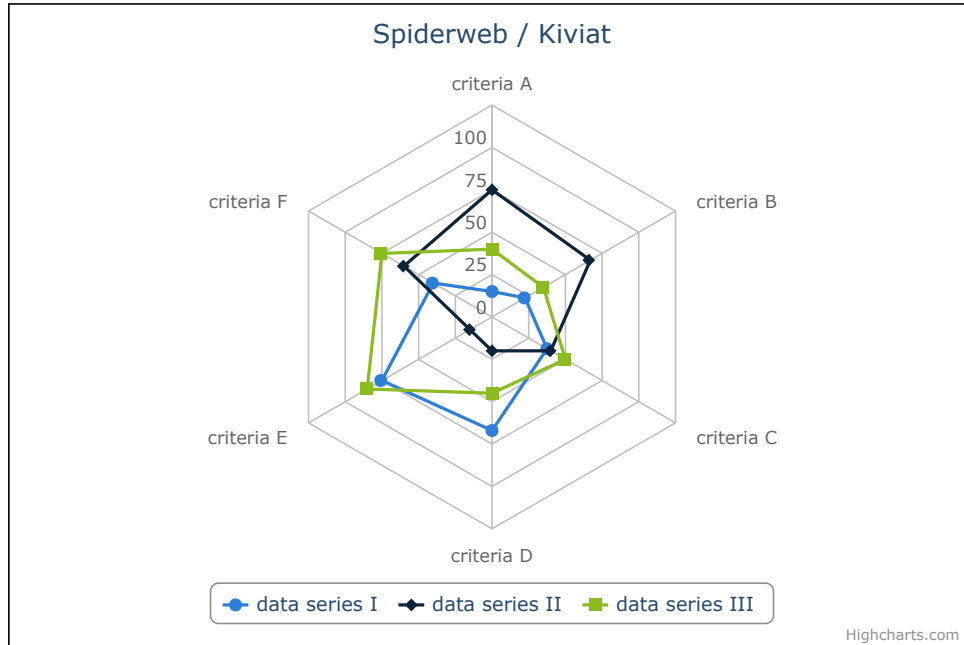


Figure 4.8.: Example for a kiviat chart.

For a kiviat chart as shown in Figure 4.8, the x-axis has to consist of discrete categories whereby every category corresponds to one spoke in the diagram. Though kiviat charts are often used in dashboards [Burger, 2006; Kerzner, 2013; Tiarks, 2008], experts criticize them as hard to decode and often unnecessary [Few, 2009; Kerzner, 2013]. Usually the same data can be represented with a line or column chart without losing information as shown in Figure 4.9 . The sole exceptions are graphs with different quantitative scales, data fitting a circular display or graphs with the objective to represent symmetry instead of magnitude [Few, 2005].

Name:	kiviat chart
X-Axis:	categories
Y-Axis:	linear or logarithmic
No. of Metrics:	multiple
Purpose:	snapshot

Table 4.4.: Properties of kiviat charts.

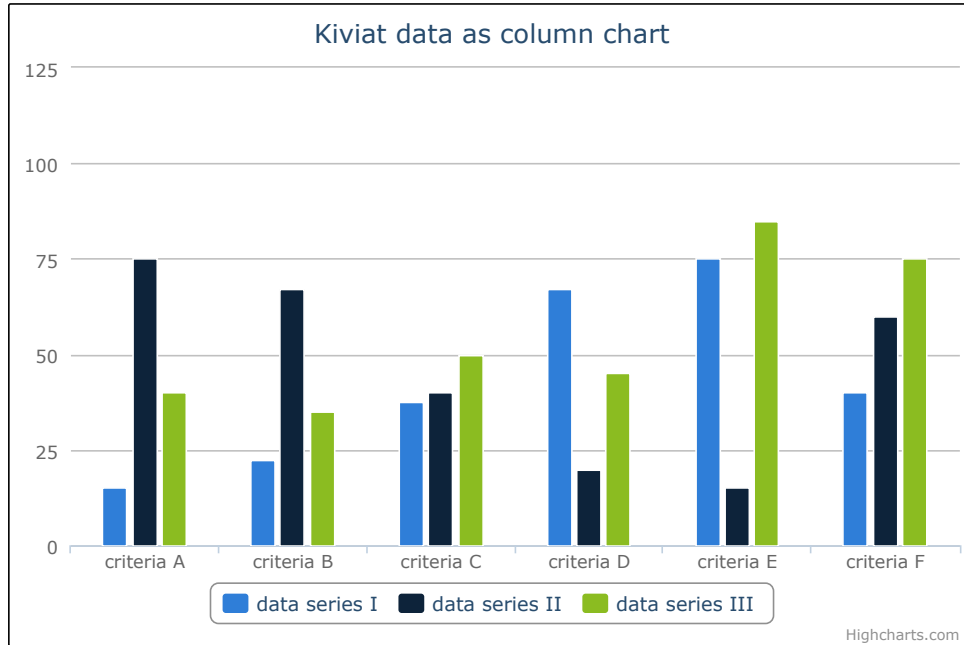


Figure 4.9.: Kiviati chart of Figure 4.8 as column chart.

4.5. Bullet chart

In business dashboards, gauge visualizations as shown in Figure 4.10 are often utilized to display the current, snapshot value of a single, important metric. Though often used in the industry [Kerzner, 2013; Parmenter, 2010; Rasmussen et al., 2009], visualization experts reject them for being hard to decode and compare, and for wasting a great deal of space [Cleveland, 1985; Few, 2006; Tufte, 2001]. Therefore, Stephen Few developed the so called *bullet chart* to address the need for a suitable visualization for the display of a single metric, specifically for dashboards [Few, 2006].

A bullet chart consists of a bar that encodes the value of the metric, a quantitative scale for easier decoding, background fill colors to encode qualitative ranges like bad, satisfactory, and good, and a symbol marker to encode a comparative measure [Few, 2006]. Due to its linear design, bullet charts can be oriented either horizontally as shown in Figure 4.11 or vertically, and several charts can be placed next to each other in a relatively small space.

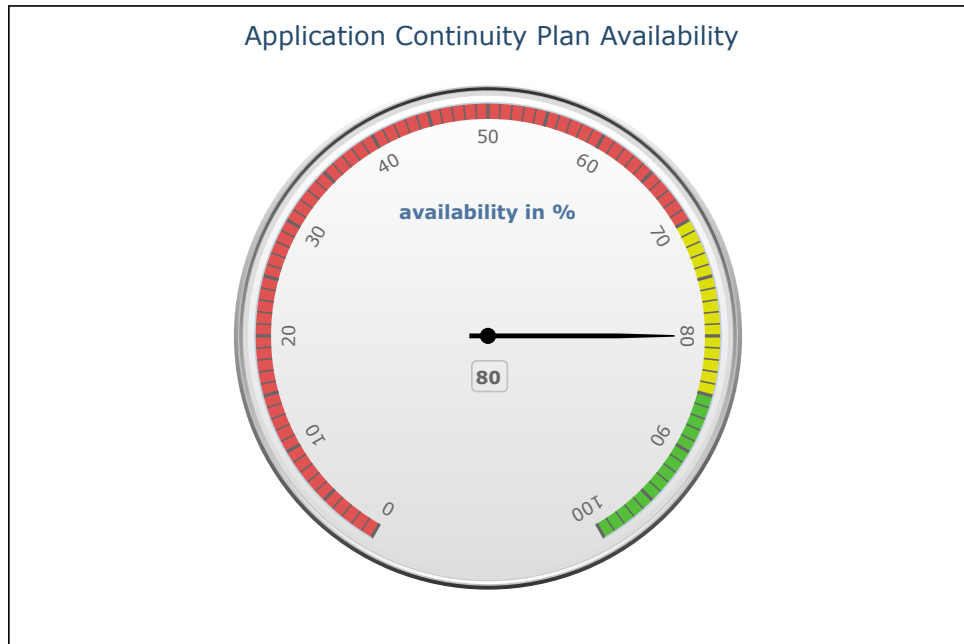


Figure 4.10.: Example for a single metric visualized as gauge.

For the visualization of EAM metrics, bullet charts can be utilized to provide a snapshot view of a single metric next to a target value.

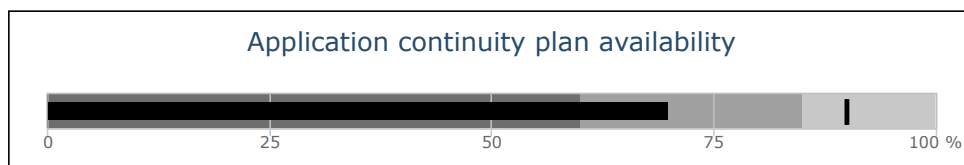


Figure 4.11.: Example for a bullet chart according to Few [2006].

4.6. Critical reflection of the proposed visualization types

The selected visualization types support all metric visualization purposes as outlined in Section 3.3, that is all metrics of the Metric Catalog can be visualized alone, in combination, as a snapshot as well as over time. Table 4.6 shows a recommendation on which visualizations to use for which metric result type, based on the findings of Chapter 3. For the illustration of metric values over time, line charts and column charts are equivalent.

Name:	bullet chart
X-Axis:	N/A
Y-Axis:	linear
No. of Metrics:	one
Purpose:	snapshot

Table 4.5.: Properties of bullet charts.

However, for the visualization of multiple metrics, the focus of column charts is more on the value differences between the different metrics, whereas line charts focus more on the value changes over time.

Result type	Perspective	
	Snapshot	Over time
monetary value	bullet chart	line / column chart
multiple values	column chart	line / column chart
number	bullet chart	line / column chart
percentage	bullet chart	line / column chart

Table 4.6.: Recommended use of visualization types for the result types of metrics.

All visualization types were chosen due to their relevance in literature, Table 4.7 shows where the specific visualization types are mentioned and utilized.

A special case are gauges and bullet charts. Though gauges are mentioned several times in literature (e.g. by Kerzner [2013], Parmenter [2010], and Rasmussen et al. [2009]) they were dropped in favour of bullet charts as defined by Few [2006], because Cleveland [1985], Tufte [2001], and Few [2006] make compelling arguments against their use. Pie charts and kiviart charts however were kept due to the former being very popular in industry (cf. Table 4.7) and the latter having their legitimate use in certain situations [Few, 2005].

In addition to that, only quantitative visualizations were considered as opposed to qualitative visualizations (e.g. traffic lights, smileys, arrows) that are often used in the industry [Kerzner, 2013; Rasmussen et al., 2009]. Instead of e.g. traffic lights, colored bullet charts can be used to provide a qualitative view of a metric.

4.6. Critical reflection of the proposed visualization types

Figure 4.12 and Figure 4.13 show a metric visualized as traffic light and as bullet chart respectively. In both visualizations, it is clearly visible that the value of the metric is considered *yellow*, but the bullet chart additionally conveys the current and the target value of the metric, and how close the value is to being considered *green*. A colored bullet chart therefore transports more information without being less “*readable*”, though Few recommends using color intensity variation instead of different colors to take color-blind readers into consideration [Few, 2006].

Visualization type	Literature
Line chart	[Botthof et al., 2008], [Burger, 2006], [Cleveland, 1985], [Few, 2009], [Kerzner, 2013], [Parmenter, 2010], [Rasmussen et al., 2009], [Tiarks, 2008]
Column chart	[Botthof et al., 2008], [Burger, 2006], [Cleveland, 1985], [Few, 2009], [Kerzner, 2013], [Parmenter, 2010], [Rasmussen et al., 2009], [Tiarks, 2008]
Pie chart	[Botthof et al., 2008], [Burger, 2006], [Rasmussen et al., 2009], [Tiarks, 2008]
Kiviat chart	[Burger, 2006], [Kerzner, 2013], [Tiarks, 2008]
Bullet chart	[Few, 2006]

Table 4.7.: Visualization types as found in literature.

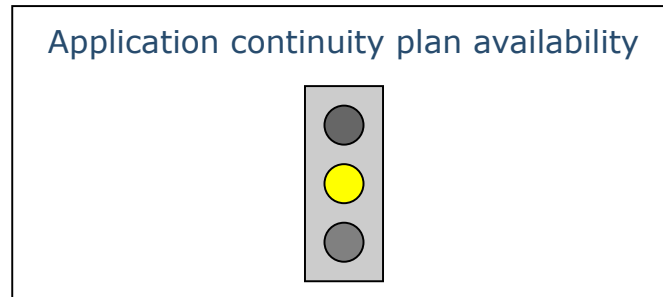


Figure 4.12.: Example for a traffic light.

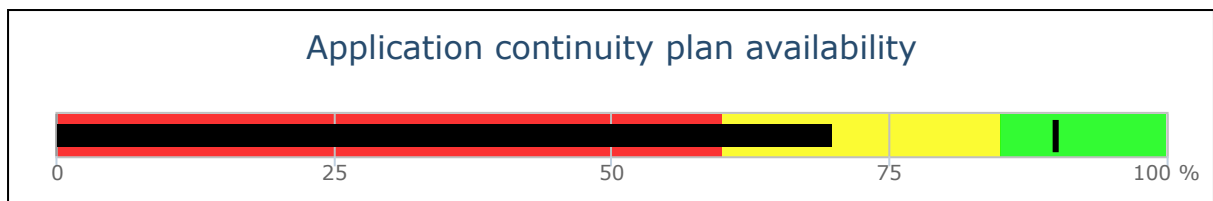


Figure 4.13.: Example for a colored bullet chart.

5. Prototypical implementation

5.1. Tricia

As stated in requirement 3, Section 3.4, one of the foundations for the implementation will be Tricia, a java-based wiki-system aimed at enterprise users. It was developed by the sebis chair of the Technical University of Munich and ownership was later transferred to the software company infoAsset. Tricia's main objective is the provision of collaborative information management based on a flexible data model [Büchner et al., 2010]. Due to its hybrid wikis and data model, Tricia can additionally be used as an EAM tool [Matthes and Neubert, 2011].

The following sections provide a short introduction into Tricia's architecture and wiki system to explain the context of the planned metric visualizations.

5.1.1. Architecture

Tricia employs the Model-View-Controller (MVC) pattern as an architectural guideline to decouple the user interface (view) from the web application's data (model). The application's logic (controller) thereby acts as a custodian in between. The pattern ensures the separation of responsibilities within the application and fosters the reuse of models and views.

Models in Tricia

Tricia stores its data in an arbitrary database system, but utilizes a flexible persistence layer to abstract from the concrete database implementation. The result is that the model layer is independent of the underlying database system.

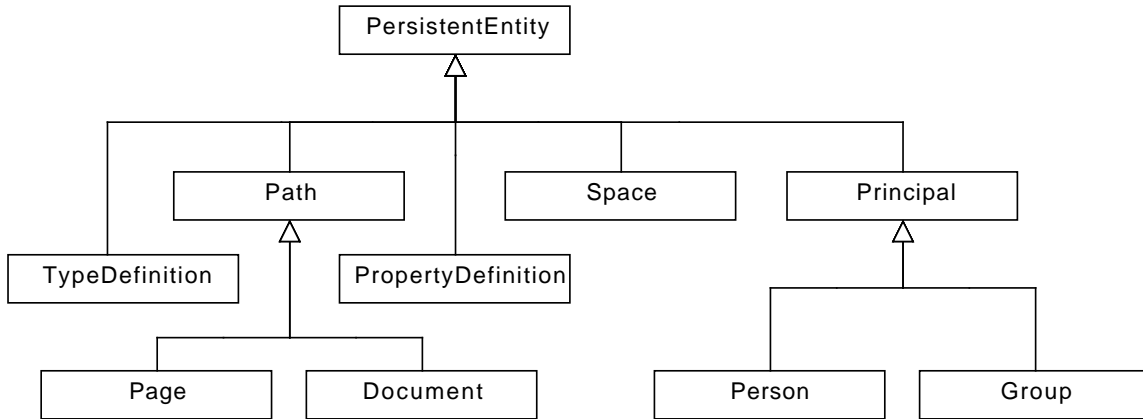


Figure 5.1.: Basic model hierarchy of Tricia according to Reschenhofer [2013].

Figure 5.1 shows the basic models that are predefined in Tricia and their hierarchy. Those models are extensible by derivation from one of the existing model classes and are as follows [Reschenhofer, 2013]:

TypeDefinition & PropertyDefinition: *TypeDefinitions* and *PropertyDefinitions* allow the definition of an information model’s schema at runtime, they are therefore called schema objects. Each *TypeDefinition* thereby contains an arbitrary number of *PropertyDefinitions* and can be assigned to information objects (e.g. pages, documents).

Page & Document: *Pages* and *Documents* are the main information objects and consist at least of a name, tags, a type, attributes, and relations. *Pages* additionally contain unstructured rich-text content, whereas *Documents* represent a file uploaded to Tricia. Attributes and relations are either defined by the according *TypeDefinition* (and its *PropertyDefinitions*) or defined as free attributes, that is arbitrary name-value-pairs. Due to each page having structured (type, attributes, relations) and unstructured (rich-text) data, they are called *Hybrid Pages*.

Space: *Spaces* are containers for pages, documents, type definitions, and property definitions. They are in their nature comparable to java packages, whereas each space defines its own namespace, that is two spaces may have types of the same name, but within each space the type name has to be unique. In addition to that Tricia supports the im- and export of whole spaces and their objects.

Person & Group: *Person* and *Group* represent users and user groups respectively. Authentication is done by providing a username and password and grants a user or user group either read-only access to certain Tricia objects, or read and write access, depending on the rules set for the object.

Controllers and Views in Tricia

According to Reschenhofer [2013] Tricia, as a web application employing the MVC pattern, handles HTTP requests as shown in Figure 5.2:

1. The web server is responsible for the creation or restoration of sessions and the user authentication.
2. The web server forwards the request to the responsible handler (controller) based on the request URL.
3. The handler checks, whether the current user is allowed to perform the requested action.
4. The handler retrieves appropriate models and performs the associated business logic (e.g. model update), if the user is properly authorized.
5. Finally, the handler prepares the view, which has to be delivered to the requesting client. This view might be based on a template defining layout and design of the view, its content is instantiated by the view itself.

5.1.2. Hybrid wikis

In Tricia, the so-called *hybrid wikis* are one of the fundamental concepts. Hybrid thereby refers to an emergent enrichment of unstructured content (e.g. rich-text documents) with structure (types, attributes, and relations) [Neubert, 2012]. As stated in Section 5.1.1, two of the base models in Tricia are *TypeDefinitions* and *PropertyDefinitions*, whereby a type definition consists of several property definitions, which in turn may define certain integrity rules:

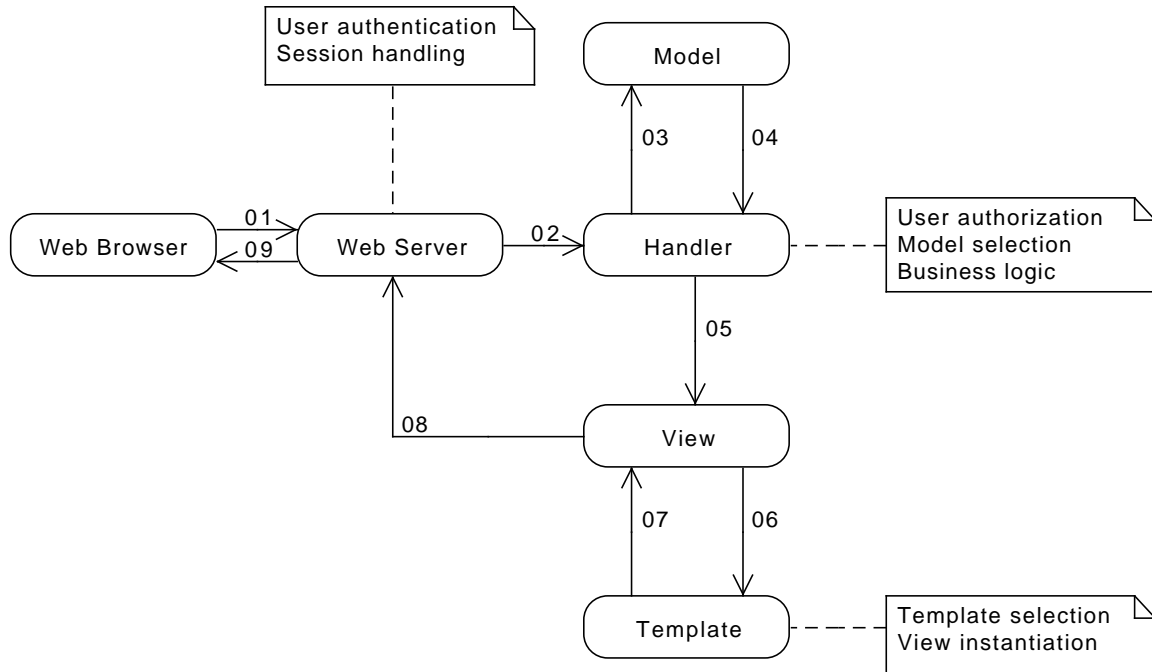


Figure 5.2.: Basic processing of a HTTP request to Tricia according to Reschenhofer [2013].

Type: If a type is defined for an attribute, the attribute value of each type definition’s instance has to be of this type. Tricia provides a basic set of attribute types, e.g. *Text*, *Number*, *Date*, *Boolean*, and *Reference* (a relation to other instances, optionally restricted to a specific type definition). For example in Figure 5.3, the attribute *covered by* is of type *Reference* whereas the linked instance has to be of type *IT continuity plan*.

Multiplicity: The attribute of each of the type definition’s instances has to have the number of values as defined by the property definition. For this purpose Tricia provides a set of predefined multiplicities (*Any number*, *At least one*, *Exactly one*, and *Maximal one*). For example in Figure 5.3, each instance of *Business Application* has to provide at most one value for the property definition *covered by*.

The relationship between the schema objects (TypeDefinition and PropertyDefinition) and the information objects (Pages) is shown in Figure 5.4.

Attributes of type **Business Application**

Attribute Name	Attribute Type	Multiplicity
Name	Text	Exactly one value
covered by	Reference IT continuity plan	At most one value
is Critical	Boolean	Exactly one value

Figure 5.3.: Exemplary *TypeDefinition* “*Business Application*” consisting of several *PropertyDefinitions*.

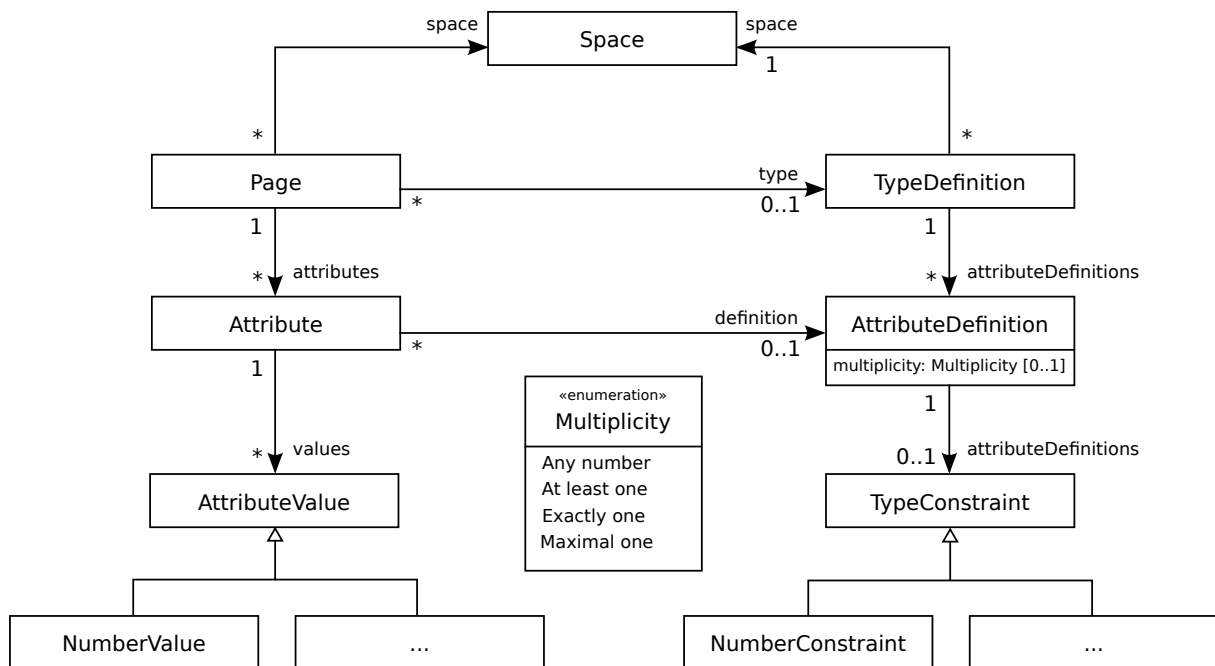


Figure 5.4.: Abstract hybrid wiki data model according to Reschenhofer [2013].

5.1.3. Block Substitutions

In addition to regular rich-text content, Tricia’s rich-text properties (like the content of a wiki page) support so-called *BlockSubstitutions*, that is executable blocks inside the HTML markup. During editing of the property, a block’s definition is shown, while upon viewing the property Tricia executes the block and substitutes its definition with the result of the execution. Listing 5.1 shows an example for the BlockSubstitution *metricVisualization*, as defined by the implementation, allowing to embed the visualization configuration inside the HTML markup. Upon viewing the block is replaced with the actual code for the visualization.

```
1 <span class="embeddedMetricVisualization">
2   $[metricVisualization()]$
3   {/* visualization configuration */}
4   $metricVisualization]$
5 </span>
```

Listing 5.1: Example for a *BlockSubstitution*.

5.2. MxL 2.0

As stated in Section 2.2 Monahov et al. designed a DSL capable of formally defining and calculating all metrics of the Metric Catalog. This language was inspired by both the Object Constraint Language (OCL) [Object Management Group (OMG), 2012] and Microsoft’s LINQ [Microsoft, 2013]. While the language itself was called MxL 1.0, it was later refined to version 2.0, the implementation in Tricia was called TxL 1.0. Its main properties were functional programming, object-orientation, sequence-orientation, a dynamic type system, and dynamic bindings [Reschenhofer, 2013].

Though MxL 1.0 was already able to define all metrics of the Metric Catalog, it had some distinct shortcomings, e.g. no compile-time analysis of MxL expressions and no arithmetic or comparison operations. To address this and more concerns, Thomas Reschenhofer developed a second iteration of MxL in his master’s thesis [Reschenhofer, 2013], called MxL 2.0. The corresponding implementation in Tricia is therefore called TxL 2.0.

```
1  /* Determine all critical applications */
2  let criticalApplications =
3    find('Business Application').where('is critical') in
4
5  /* Calculate proportion of covered critical applications */
6  criticalApplications.ratio('covered by' <> null)
```

Listing 5.2: The metric *Application continuity plan availability* defined as MxL expression [Reschenhofer, 2013].

Currently, all metrics defined by the Metric Catalog are implemented as so-called MxL 2.0 custom functions, Listing 5.2 shows an example metric. The implementation as presented in Chapter 6 uses this MxL expressions to calculate the values needed for the visualizations.

5.3. Identification & comparison of existing visualization engines

The following section provides an overview and evaluation of different possible visualization solutions for the implementation. Due to the nature of Tricia, all candidates are libraries and technologies suitable for the web, i.e. they work in the browser. Important features are in no particular order:

- Support of the selected visualization types (Chapter 4).
- Dynamic generation of visualizations (no pre-generated images).
- Documentation available.
- Optional: Interactivity, e.g. tool tips, filtering, etc.

5.3.1. HTML5 Canvas

Canvas is a HTML element proposed for HTML5. It is standardized by the W3C⁷ and provides a resolution dependent bitmap canvas, which can be used to render graphics on the fly in the browser [World Wide Web Consortium (W3C), 2014]. The content of

⁷World Wide Web Consortium

the canvas can be manipulated with JavaScript calls, however HTML5 canvas itself is very low-level and only provides the means to draw basic shapes and paths. Furthermore the canvas is raster-based in contrast to vector-based formats like SVG⁸ and has to be redrawn for every change.

Plain HTML5 Canvas is not suited for the visualization of metrics, because every element in the visualization would have to explicitly be drawn (e.g. single rectangles for the bars in a bar chart). Hence several JavaScript libraries were built on top of the basic canvas functionality to allow the easy creation of data visualizations. Two of those libraries are *Chart.js*⁹ and *RGraph*¹⁰. Since Chart.js only provides six different visualization types and is lacking essential features like data labels on pie charts, only RGraph is considered in this evaluation.

The results of the evaluation are shown in Table 5.1. As can be seen, except for bullet charts all selected visualization types are supported. Though new visualizations can be dynamically generated, it is hardly possible to follow best-practice guidelines found in literature. For example it is not even possible to label the axes. Furthermore, RGraph offers no interactivity in terms of tool tips for single data points or filtering of values.

Visualization Type	Supported	Requirement	Supported
Line chart	✓	Dynamic generation	✓
Area chart	✓	Documentation available	✓
Column chart	✓	Interactivity	–
Bar chart	✓		
Pie chart	✓		
Kiviat chart	✓		
Bullet chart	–		

Table 5.1.: Evaluation of RGraph

⁸Scalable Vector Graphics

⁹<http://www.chartjs.org>

¹⁰<http://www.rgraph.net>

5.3.2. Raphaël / g.Raphaël

Raphaël¹¹ is a JavaScript library for the manipulation of vector graphics based on the SVG format. It abstracts from the specific browser implementation, but only offers functionality similar to HTML5 Canvas, that is it can draw and manipulate basic shapes and paths. Therefore it is on its own not suited for the visualization of metrics, too. Hence *g.Raphaël*¹² was built on top of Raphaël to provide easy data visualization capabilities.

The evaluation of g.Raphaël is shown in Table 5.2. As you can see, it only supports four of the seven selected visualizations, though it offers at least some interactivity for those (value tool tips). A comprehensive documentation is available, but g.Raphaël lacks important features to make the visualizations follow best-practice recommendations found in literature. Among other things it is not possible to label axes or data points and to display grid lines for easier deciphering of line charts.

Visualization Type	Supported	Requirement	Supported
Line chart	✓		
Area chart	—		
Column chart	✓	Dynamic generation	✓
Bar chart	✓	Documentation available	✓
Pie chart	✓	Interactivity	✓ ^a
Kiviat chart	—		
Bullet chart	—		

^aTool tips

Table 5.2.: Evaluation of g.Raphaël

5.3.3. D3.js / NVD3

D3.js¹³ is a JavaScript library for manipulating web documents (HTML, SVG, CSS) based on data. Similar to HTML5 Canvas and Raphaël the visualization capabilities of D3.js

¹¹<http://www.raphaeljs.com>

¹²<http://g.raphaeljs.com>

¹³<http://d3js.org>

are very low-level and mostly consists of simple shapes, paths, and combinations thereof. It is therefore on its own not suited for the visualization of metrics. Just as with HTML5 Canvas and Raphaël there is a library built on top of D3.js specific for the visualization of metrics, *NVD3*¹⁴.

The evaluation results for NVD3 are shown in Table 5.3. First of all, NVD3 is a very young library and still in beta stage. This shows especially in the fact that there is no documentation available yet. The lack of documentation makes it hard to gauge the full capabilities of the project, but according to the examples available on the web site at least six of the seven selected visualizations are supported. Visualizations can be designed interactive with tool tips, filtering and a navigation bar.

Visualization Type	Supported	Requirement	Supported
Line chart	✓		
Area chart	✓		
Column chart	✓	Dynamic generation	✓
Bar chart	✓	Documentation available	–
Pie chart	✓	Interactivity	✓
Kiviat chart	–		
Bullet chart	✓		

Table 5.3.: Evaluation of NVD3

5.3.4. Highcharts / Highstock

Highcharts and *Highstock*¹⁵ are JavaScript libraries for the creation of charts. They offer the same basic feature set, but Highstock provides additional visualization types focused on the display of financial data. In the following chapters of the thesis Highcharts is used synonymous for both. The libraries are based on vector graphics (SVG) in modern browsers and have HTML5 Canvas as a fallback option for browsers not supporting SVG.

¹⁴<http://nvd3.org/>

¹⁵<http://www.highcharts.com/>

Visualizations are defined on a higher level than with D3.js or Raphaël and can be created from a configuration string written in JSON¹⁶.

The evaluation results for Highcharts are shown in Table 5.4. As can be seen, established visualizations are well supported, kiviatic charts are just line or area charts with polar axes and bullet charts can be designed as bar charts. All visualizations are dynamically generated and can be configured to follow best-practice recommendations found in literature. A comprehensive documentation is available and visualizations can be made interactive to a certain degree with filtering options, custom tool tips, and navigation bars.

Visualization Type	Supported	Requirement	Supported
Line chart	✓		
Area chart	✓		
Column chart	✓	Dynamic generation	✓
Bar chart	✓	Documentation available	✓
Pie chart	✓	Interactivity	✓ ^c
Kiviatic chart	✓ ^a		
Bullet chart	✓ ^b		

^aAs a polar line or area chart.

^bAs a specially designed bar chart.

^cFiltering on series, tool tips, navigation bar.

Table 5.4.: Evaluation of Highcharts

5.3.5. Critical reflection

Table 5.5 shows an overview over the evaluation results. In the table a ● means that the specific feature is available, a ◐ means, that it is partially available, and a ○ means that it is not available. The explanation for partial availability is in the corresponding section.

Both RGraph and g.Raphaël lack essential visualization types and only offer partial to no interaction with the visualizations. They are therefore not suited for this thesis.

¹⁶JavaScript Object Notation, <http://www.json.org/>

Overall Highcharts and Highstock are the best candidates, though bullet charts are not supported as native visualization type, but have to be designed as customized bar chart. For the future, NVD3 looks very promising, but its alpha-status and the lack of a documentation make it not suited for the thesis.

	RGraph	g.Raphaël	NVD3	Highcharts
Line chart	●	●	●	●
Area chart	●	○	●	●
Column chart	●	●	●	●
Bar chart	●	●	●	●
Pie chart	●	●	●	●
Kiviat chart	●	○	○	●
Bullet chart	○	○	●	◐
Dynamic	●	●	●	●
Documentation	●	●	○	●
Interactivity	○	◐	●	●

Table 5.5.: Visualization engine evaluation results.

5.4. Iterative prototype development

The following section details the architectural progression from the first prototype to the final implementation. Since the implementation was developed following an evolutionary prototyping model [Davis, 1992], each iteration wasn't planned in detail with a target architecture and a complete specification document, but rather developed in a very experimental manner to better understand the requirements and possibilities with each iteration. This approach leads to distinct changes in the architecture over time, with the final implementation having very little in common with the initial prototype.

The basic idea was to built a configuration interface in HTML, using JavaScript to dynamically load the interface's content, to allow the user to configure basic aspects of the visualization, i.e. the type (line chart, column chart, etc.), the data sources (which metrics to display), the axis' types and ranges, and some style options (title, subtitle, labels,

etc.). The resulting configuration was to be embedded in the page and the visualization rendered when the page is viewed.

Despite the ever-changing architecture, two properties stayed the same: Visualizations can be added with the existing page editor in Tricia and they are embedded in a page's rich-text content as *BlockSubstitution* as specified in Section 5.1.3. Visualizations can therefore be displayed along the underlying MxL expressions as defined by Reschenhofer [2013].

5.4.1. First Prototype

The first prototype was built as a proof-of-concept to show that Highcharts is a viable solution for the visualization of metrics defined as MxL expressions. Therefore a plugin for Tricia's page editor (TinyMCE¹⁷) was developed to provide the user with the means to call the visualization configurator while editing the rich-text content of a page and to embed the resulting Highcharts configuration as JSON-string within the page.

With the option in mind to filter the available configuration options based on the selected visualization sometime later, all configuration options were stored on the server-side along with an empty generic visualization template. This led to the following components, with *Handlers* being controller components as stated in Section 5.1.1 and the *Block Substitution Template* being the server-side code for the *metricVisualization* block substitution as defined in Section 5.1.3 as well as the HTML template the block is replaced with:

Dialog Handler This handler was called with the selected visualization as parameter (if there was any) when the configurator was launched. It rendered the configuration dialog and provided an empty visualization template if none was given.

Style Handler This handler was called with the current visualization as parameter. It provided a list of possible x- and y-axis types respectively and a list of possible intervals for date axes (e.g. *monthly*, *quarterly*, *annually*, etc.).

Type Handler This handler was called with the current visualization as parameter. It provided a list of possible visualization types.

¹⁷<http://www.tinymce.com/>

Block Substitution Template Upon viewing a page with an embedded visualization configuration, the block substitution template calculated the data for the according metrics and added the resulting value-pairs to the visualization. The HTML template contained some JavaScript to render the actual visualization.

Client-side JavaScript In the first prototype the JavaScript was an unstructured collection of functions to add dynamic behavior to the configuration dialog, this includes adding values to the configuration dialog (input fields, type and date selections) and hiding certain elements depending on the selected values (e.g. date inputs when no date axis is selected). On top of that it stored the configuration as block substitution on the page.

Available visualization types Line/area chart, column/bar chart, pie chart.

Shortcomings

- Kiviat and bullet charts were unsupported.
- No error handling whatsoever.
- No built-in rules for meaningful visualizations (e.g. column charts with logarithmic scaling were not prevented).
- Application logic split between client and server.
- Complex visualizations could not be predefined.
- Very unstructured client-side.

5.4.2. Second Prototype

The second prototype was built after realizing that filtering down the configuration options can be done on the client-side, too. This removed the now defunct style and type handler and the empty visualization template from the dialog handler. Available date intervals were kept on the server-side, because the substitution template needed the information, and otherwise it would have been stored redundant.

To reflect these changes, a date interval handler was added, providing possible date intervals, and the configuration options and the visualization templates were moved to the client-side. This stage consisted of the following components:

Dialog Handler This handler was called without parameters when the configurator was launched and rendered the configuration dialog.

Date interval Handler This handler was called without parameters to get a list of possible date intervals.

Block Substitution Template Upon viewing a page with an embedded visualization configuration, the block substitution template calculated the data for the according metrics and added the resulting value-pairs to the visualization. The HTML template contained some JavaScript to render the actual visualization.

Client-side JavaScript The second iteration brought more structure into the collection of functions. Additionally the empty visualization template and all configuration options with the exception of date intervals were now readily available without additional server calls. Apart from that it stayed mostly the same.

Visualization types Line/area chart, column/bar chart, pie chart

Improvements over the first prototype

- More application logic on the client-side.
- More structure on the client-side.

Shortcomings

- Kiviatic and bullet charts were still unsupported.
- Still no error handling.
- Still no built-in rules for meaningful visualizations (e.g. column charts with logarithmic scaling were not prevented).
- Application logic still split between client and server.

- Complex visualizations could still not be predefined.

5.4.3. Third Prototype

The third prototype was an attempt to move most of the application logic to the client-side and to lay a foundation for error handling. This removed all server interaction during the configuration process, which would otherwise possibly stall when the server took some time to respond with the configuration options. Therefore two new handlers were introduced to retrieve information about a single MxL expression (its input types) and to calculate a list of input-output value pairs for a given MxL expression and given input.

At this point all application knowledge was on the client-side, among other things the server had now knowledge about the structure of a visualization configuration anymore. This got especially useful while building the final iteration.

Dialog Handler This handler was called without parameters when the configurator was launched and rendered the configuration dialog. (No changes)

Info Handler This handler could be called with the name of an MxL expression and provided the number and types of its input parameters. Thereby the handler tried to find the expression in the current page space and then, if it wasn't defined there, in the global space. It served as a foundation for possible error handling in terms of type checking the metrics for the visualization.

Data Handler This handler could be called with the name of a an MxL expression, the current page id, and a list of string or date values. Each item of the list was used as input to the specified expression to calculate a numeric value. The resulting input-output value pairs were returned as data for the visualization. Thereby the handler tried to find the expression in the current page space and then, if it wasn't defined there, in the global space.

Block Substitution Template Upon viewing a page with an embedded visualization configuration, the block substitution didn't calculate the missing values of the visualization on the server side anymore, but passed the visualization configuration

to the client. The JavaScript code in the HTML template then calculated the data series with the aid of the data handler and rendered the actual visualization.

Client-side JavaScript The third iteration brought new functions for the calculation of data series with the aid of the data handler and further refined the functions into smaller, more reusable pieces. On top of that, all configuration options were now provided on the client-side. However, the addition of kiviatic charts introduced some problems. Since Highcharts treats kiviatic charts as a special case of line charts and until then the visualization type was a one to one mapping to the types offered by Highcharts, special cases had to be introduced in the code to properly recognize and handle kiviatic charts. At this point, it was clear that every new visualization would have to be treated as an exception in the code, with all its properties hard-coded somewhere. This would have severely hindered further development and made the addition of bullet charts extraordinarily costly.

Visualization types Line/area chart, column/bar chart, pie chart, kiviatic chart.

Improvements over the second prototype

- All application logic on the client-side.
- More reusable components on the client-side.
- Support of kiviatic charts.
- Foundation for error handling.

Shortcomings

- Bullet charts still unsupported.
- Still no error handling.
- Still no built-in rules for meaningful visualizations (e.g. column charts with logarithmic scaling were not prevented).
- Complex visualizations could still not be predefined.
- Impossible to add new visualizations without adding exceptions to the code.

5.4.4. Final Architecture

The findings and limitations from the third prototype led to the development of the final architecture, which is mostly a major rewrite of the client-side JavaScript code. It is now structured in an object-oriented manner and dropped the generic visualization template in favour of more extensive individual templates for each visualization type. This leads to more options in the definition of each visualization, a finer control over the configuration made by the user, and no treatment of edge cases in the code. New visualization types can now be added without changing the code.

Server-calls are entirely stateless and all information needed has to be encoded in the URL. Furthermore, they were reduced to a minimum to speed up the application if the server is only slowly responding. This chapter provides an overview over the structure of the application, separated into client-side and server-side components, the individual visualization templates and the configuration process is subject in Chapter 6.

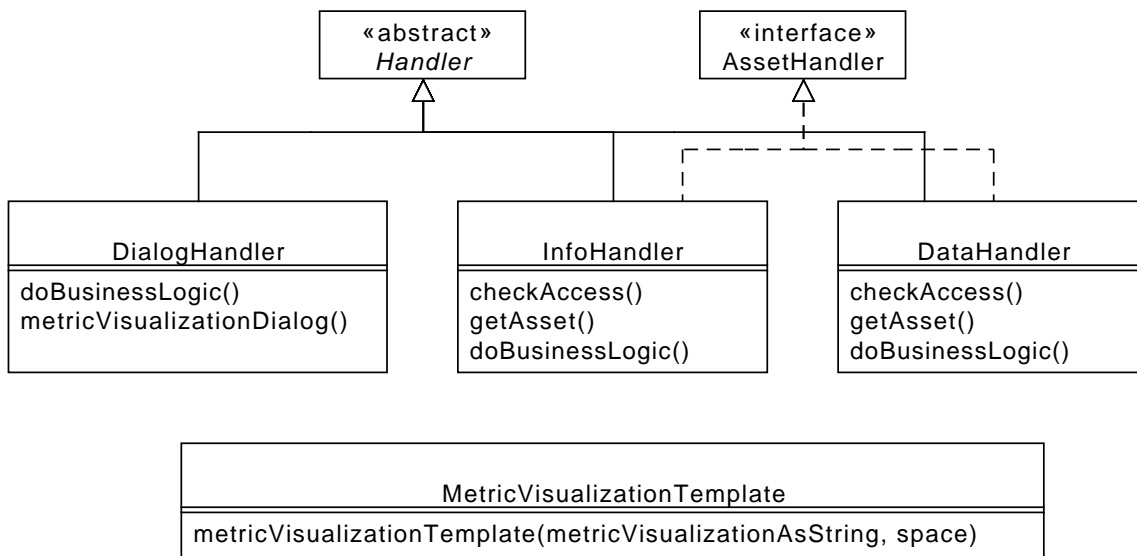


Figure 5.5.: Class diagram of the configurator backend. (Private attributes omitted)

Figure 5.5 shows an UML class diagram depicting the server-side classes. Note that the name-prefix *Metric* is used for the template to avoid name conflicts.

All handler classes have to subclass the abstract class *Handler* defined by Tricia, the

5.4. Iterative prototype development

info handler and data handler class have to additionally implement the *AssetHandler* interface, because they need access to MxL expressions and have to check whether the user is properly authorized to access those expressions. The four classes fulfill the same roles as in the third prototype. Unfortunately due to the way Tricia works, the parameters for every handler call are not reflected in the diagram:

DialogHandler Stateless handler, provides the configuration dialog HTML template.

InfoHandler Stateless handler, provides the number and type of parameters for a given MxL expression.

DataHandler Stateless handler, provides calculated input-output value pairs for a given MxL expression with a given list of input parameters.

MetricVisualizationTemplate Stateless block substitution template, fills the HTML template with the visualization configuration.

Figure 5.6 shows a class diagram for the client-side JavaScript code. The client-side code was divided into five classes to separate concerns and encapsulate all dialog-specific code (*data binding*) and all server-interaction:

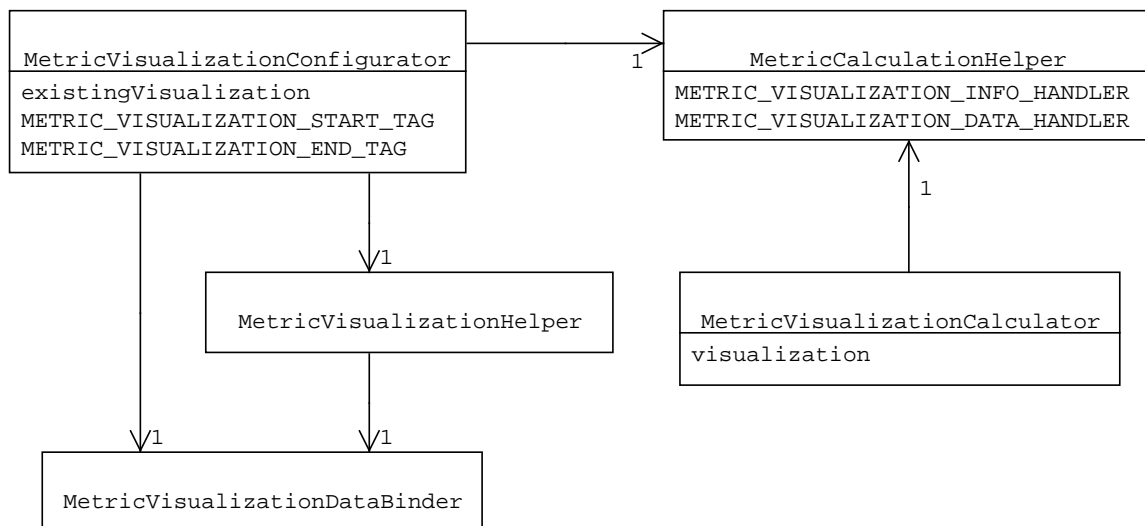


Figure 5.6.: Class diagram of the visualization configurator. (Methods omitted)

MetricVisualizationConfigurator Main class for the configuration dialog. An instance is created whenever a visualization is to be added or edited with the configuration as parameter in the latter case. It coordinates the filling of the available input fields and enforces the constraints set in the visualization template. In the process, it has no direct knowledge of the configuration dialog itself (e.g. HTML element ids and classes), but employs an instance of the *MetricVisualizationHelper* and the *MetricVisualizationDataBinder* class respectively.

MetricVisualizationHelper This class has no internal state, that is all instances of the class behave exactly the same. It contains methods to manipulate the state of the dialog, e.g. add and remove input fields, create tabs, fill in drop-down lists, etc. The helper class employs an instance of the *DataBinder* class to access the dialog HTML elements.

MetricVisualizationDataBinder This class encapsulates most of the data binding, that is it provides methods to access the dialog HTML elements. It has no internal state and offers a wide range of methods, among other things, to change the value of input fields, display error messages, and enable and disable elements.

MetricCalculator This class is employed by the block substitution HTML template to calculate the series data for a given visualization and provide a configuration for Highcharts or provide an error message if the calculation fails (e.g. due to wrong parameter types or missing MxL expressions). In the process, it has no direct server interaction, but employs an instance of the *MetricCalculationHelper* class to retrieve information and values from the server.

MetricCalculationHelper This class has no internal state and encapsulates all interaction with the server, that is it provides methods to retrieve information about an MxL expression (number of parameters and parameter types) and to calculate the data series for a visualization.

6. Selected implementation aspects

Foundation

Figure 6.1 shows an example line chart annotated with the terminology used in the following sections. Furthermore, each annotation marks an aspect of a visualization that can be configured for every individual instance. Title and subtitle are free text fields to name the visualization and give a short description about its content. Similarly, x-axis title and y-axis title can be set freely to describe the values on the axes. X-axis type and y-axis type on the other hand have to be chosen from a fixed set of options for each visualization type. The x-axis type can either contain a date range as shown in the figure or contain discrete categories, e.g. as shown in Figure 4.3. The y-axis type denotes the scale of the axis and is either *linear* or *logarithmic*. For the data labels and the navigator can be decided whether each item is shown or not. Finally, the legend cannot be configured, but simply has an entry for every data series in the visualization and can be used to temporarily hide series.

6.1. Visualization template

As stated in Section 5.4.4, the final iteration of the development process dropped a generic visualization template in favour of individual templates. The templates are specified in JSON and allow to predefine configuration options, the number of metrics a visualization can display, and the underlying Highcharts configuration.

Listing 6.1 shows an empty visualization template. The template consists of four attributes, whereas every attribute consists of a JSON object:

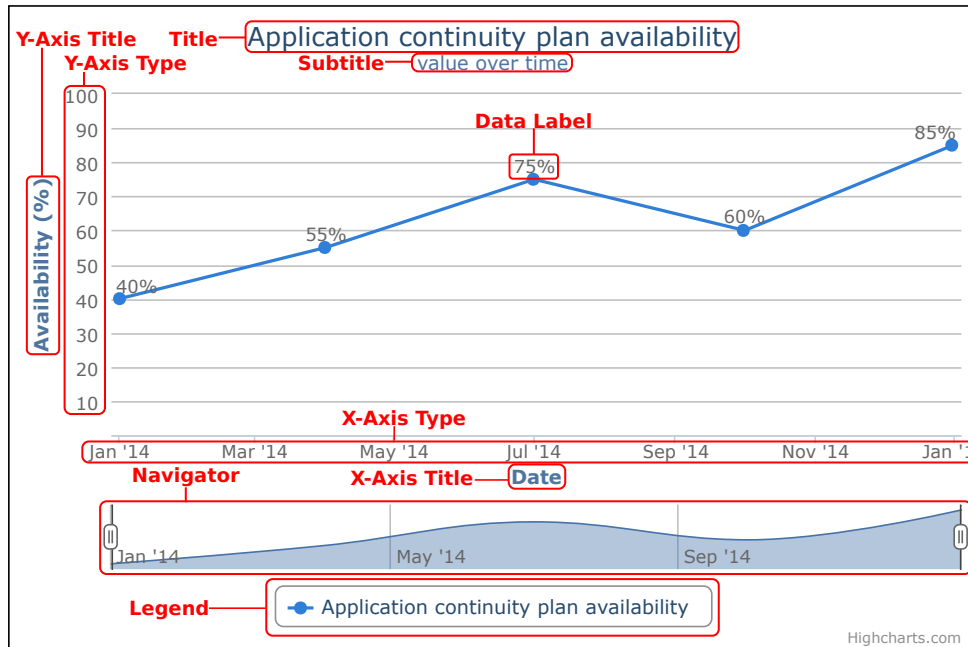


Figure 6.1.: Example visualization with annotations.

properties This attribute defines the name of the visualization, its default configuration, and its configuration constraints, e.g. whether the visualization is suited for more than one metric, which fields can be edited, and which axis types are available. Since the *name* entry is the unique identifier of a visualization template, it is embedded in the Tricia page.

configuration The configuration attribute defines an entry for every field customizable in the configuration dialog. In the visualization templates those entries are empty and filled with the configured values after the configuration process. This attribute is also embedded in the Tricia page.

highcharts_config This attribute contains the underlying JSON configuration for the Highcharts library whereby all options of Highcharts are allowed, except for predefined data series. In addition to that, the appropriate entries for the configuration options (e.g. title, subtitle, axis types, etc.) have to be present, otherwise the rendering will fail. The embedded configuration hides this attribute from the user, as it will get very long for elaborate Highcharts configurations.

```
1 {
2   "properties": {
3   },
4   "configuration": {
5   },
6   "highcharts_config": {
7   },
8   "highcharts_expert": {
9   }
10 }
```

Listing 6.1: Empty visualization template

highcharts_expert Since the configuration dialog is fixed in its selection of input fields, this attribute of the template can be utilized to further customize individual visualization instances. To that end, additional Highcharts configuration can be put into this attribute and will be merged into the *highcharts_config* element upon rendering the visualization.

Example properties attribute

Listing 6.2 and Listing 6.3 show example *properties* for a line chart and a bullet chart visualization respectively. As you can see, every property contains the unique *name* of the visualization template and the name of an example *image* if existing. Besides that, possible *types* for the x-axis and y-axis as well as possible *date intervals* can be named to set constraints for the customization options, whereas the date intervals have to be a subset of the depicted list. In addition to that, the number of metrics can be set as either *one*, e.g. for bullet charts and pie charts, or *multiple*, e.g. for line charts and column charts. This constraint is enforced during customization, too.

For every input field, default values can be defined, either to make the visualization more specific, or to aid the user. However, the default values for the titles, the axis types and the settings for data labels and a date navigator have to be set in the *configuration* section. In addition to that, every input field can be made non-editable (line 8-18), e.g. to protect default values or to clarify that no choice is possible.

```

1 "properties": {
2   "name": "Line Chart",
3   "image": "line_chart.png",
4   "x_axis_types": ["datetime"],
5   "y_axis_types": ["linear", "logarithmic"],
6   "date_intervals": ["daily", "weekly", "monthly", "quarterly", "annually"],
7   "no_of_metrics": "multiple",
8   "title_editable": true,
9   "subtitle_editable": true,
10  "x_axis_type_editable": false,
11  "x_axis_title_editable": true,
12  "y_axis_type_editable": true,
13  "y_axis_title_editable": true,
14  "data_labels_editable": true,
15  "navigator_editable": true,
16  "date_range_editable": true,
17  "date_interval_editable": true,
18  "categories_editable": true,
19  "default_categories": [],
20  "default_date_range": [],
21  "default_date_interval": "monthly",
22  "metric_series": [],
23  "metric_series_addition": {
24  }
25 }

```

Listing 6.2: Line chart properties.

The entries *metric_series* and *metric_series_addition* can be used to predefine the input fields for metrics together with custom Highcharts options per metric and to specify Highcharts options to be added to every metric entry respectively. As shown in Listing 6.3, a bullet chart needs exactly two metrics called *Measure* and *Target* whereby the first represents the actual metric to display and the second represents an MxL expression providing a target value. The number of MxL expressions needed to fill a bullet chart is therefore exactly two.

Example configuration attribute

Listing 6.4 shows an empty visualization *configuration* attribute whereas every JSON property corresponds to a field in the configuration dialog. Upon rendering of the visualization, the content of the configuration attribute is written into the existing Highcharts


```
1 "properties": {
2   "name": "Bullet Chart",
3   "image": "bullet_chart.png",
4   "x_axis_types": ["category"],
5   "y_axis_types": ["linear"],
6   "date_intervals": ["daily", "weekly", "monthly", "quarterly", "annually"],
7   "no_of_metrics": "one",
8   "title_editable": true,
9   "subtitle_editable": true,
10  "x_axis_type_editable": false,
11  "x_axis_title_editable": false,
12  "y_axis_type_editable": false,
13  "y_axis_title_editable": false,
14  "data_labels_editable": false,
15  "navigator_editable": false,
16  "date_range_editable": false,
17  "date_interval_editable": false,
18  "categories_editable": false,
19  "default_categories": [""],
20  "default_date_range": [],
21  "default_date_interval": "",
22  "metric_series": [
23    {"name": "Measure", "id": "", "data": []},
24    {"name": "Target", "id": "", "type": "scatter", "data": []}
25  ],
26  "metric_series_addition": {
27  }
28 }
```

Listing 6.3: Bullet chart properties.

configuration, whereas not every JSON property in the configuration might have an effect on the actual visualization, e.g. if the x-axis type is set to *category*, a potential date range and interval is simply ignored. The *series* array is filled with the values from the data source input fields according to potential restrictions set by the property fields *metric_series* and *series_addition*. The configuration attribute is later embedded in the Tricia page as part of a block substitution.

Example embedded configuration

As stated above and in Section 5.1.3, the information necessary to render a visualization is embedded in a Tricia Page as Tricia BlockSubstitution. The embedded configuration consists of the *name* property of the *properties* attribute, the *configuration* attribute, and the *highcharts_expert* attribute of the visualization template. An example for an

```

1 "configuration": {
2   "categories": [],
3   "date_range": [],
4   "date_interval": "",
5   "title": "",
6   "subtitle": "",
7   "x_axis_type": "",
8   "x_axis_title": "",
9   "y_axis_type": "",
10  "y_axis_title": "",
11  "navigator": true,
12  "data_label": false,
13  "series": [],
14  "container": ""
15 }

```

Listing 6.4: Empty configuration.

embedded bullet chart configuration is shown in Listing 6.5, the opening and closing tags for the block substitution are omitted. As shown in Listing 6.3 the two properties in *metric_series* were translated into two *series* properties in the configuration attribute. Here, the metric *Application continuity plan availability* is defined as measure, and an unknown expression called *applicationContinuityPlanAvailabilityTarget* supplies a target value. The resulting visualization is shown in Figure 6.2.



Figure 6.2.: Result of bullet chart configuration of Listing 6.5

As stated above, the *highcharts_expert* attribute of an visualization template can be used to add additional Highcharts configuration to a visualization instance without using the configuration dialog. It is merged with the *highcharts_config* upon rendering and therefore has to have the same structure. Listing 6.6 shows the addition of a border around the visualization as additional option to Highcharts. This feature could be used to completely change the look of a visualization instance, because it is currently not checked for violation of the constraints defined in the *properties* attribute. The resulting visualization is shown in Figure 6.3.

```
1 {
2   "properties": {
3     "name": "Bullet Chart"
4   },
5   "configuration": {
6     "categories": [""],
7     "date_range": [],
8     "date_interval": "daily",
9     "title": "Application Continuity Plan Availability",
10    "subtitle": "",
11    "x_axis_type": "category",
12    "x_axis_title": "",
13    "y_axis_type": "linear",
14    "y_axis_title": "",
15    "navigator": false,
16    "data_label": false,
17    "series": [{
18      "name": "Measure",
19      "id": "applicationContinuityPlanAvailability",
20      "data": []
21    }, {
22      "name": "Target",
23      "id": "applicationContinuityPlanAvailabilityTarget",
24      "type": "scatter",
25      "data": []
26    }
27  ],
28  "container": ""
29 },
30 "highcharts_expert": {
31 }
32 }
```

Listing 6.5: Embedded bullet chart configuration.

6.2. MxL expressions

As hinted at in Chapter 4, different visualizations fulfill different visualization duties, i.e. they need different data as input depending on their *x-axis type*. For example every data series to be displayed on a *line chart* has to exist as a list of date-value tuples, whereby every tuple represents on data point in the visualization. Listing 6.7 shows the data displayed in Figure 6.1:

Every tuple in this data series consists of a date value represented as *unix time* with a resolution of one millisecond, and the value of the metric at this point in time. Since the

```

1 {
2   "properties": {
3     "name": "Bullet Chart"
4   },
5   "configuration": {
6     /* omitted */
7   },
8   "highcharts_expert": {
9     "chart":{"borderWidth": 2}
10  }
11 }

```

Listing 6.6: Embedded bullet chart configuration with “expert” customization.

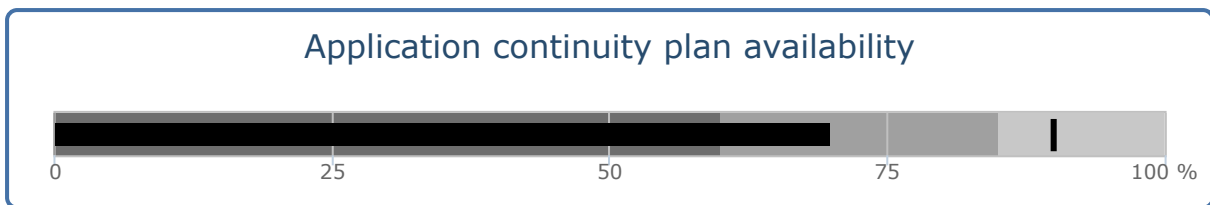


Figure 6.3.: Result of bullet chart configuration of Listing 6.6

displayed metric *Application continuity plan availability* has a *percentage* as result type, that is a *float* (e.g. 0.5), the numbers are scaled to the range between 0 and 100.

To calculate these tuples an MxL expression is needed that takes a date value as only parameter and returns a number (the value of the metric at the given date). In general, three types of expressions are supported, depending on the type of the x-axis of the visualization:

```

expression :: String → Number
expression :: Date  → Number
expression :: Number

```

That is, for visualizations with a *category* x-axis the expression has to have a *string* parameter and return a number, for visualizations with a *datetime* x-axis the expression has to have a *date* as parameter and return a number, or the expression might have no parameter and return a number for either x-axis type.

```
1  [  
2    [1388530860000,40],  
3    [1396303200000,55],  
4    [1404165600000,75],  
5    [1412114400000,60],  
6    [1419980400000,85]  
7  ]
```

Listing 6.7: Input data for the line chart in Figure 6.1.

Thus, the MxL expression representing a metric might have to be *wrapped* with another MxL expression to conform to one of the above expression types, depending on the visualization type. For example, the expression for the metric *Application continuity plan availability* as defined in Listing 5.2 has no parameter and returns a number, that is the current value of the metric. To be able to visualize the metric over time in a line chart, the metric expression has to be wrapped in another expression to have a date value as parameter and return the value of the expression at the given date. Listing 6.8 shows the wrapping MxL expression needed to calculate the data series shown above.

```
1  applicationContinuityPlanAvailability() @ date * 100
```

Listing 6.8: Wrapper expression for *Application continuity plan availability*.

At the time of writing this thesis, MxL does *not* support the calculation of historic values. The @-Operator used in Listing 6.8 is subject of current research and further explained in Section 8.1.

This wrapping is needed for all visualizations displaying either more than one metric, a metric resulting in more than one value, or displaying a metric over time. In addition to that single values (e.g. target values for bullet charts) have to be wrapped in an expression, too.

6.3. Configuration process

The visualization of metrics is divided into two parts: The configuration of a visualization and the rendering of the visualization according to its configuration. A visualization can be added to a page or edited along with other rich text content. Figure 6.4 shows an activity diagram for the configuration process. Shown are the actions performed by the user as well as those performed by the application components, separated into client-side and server-side components.

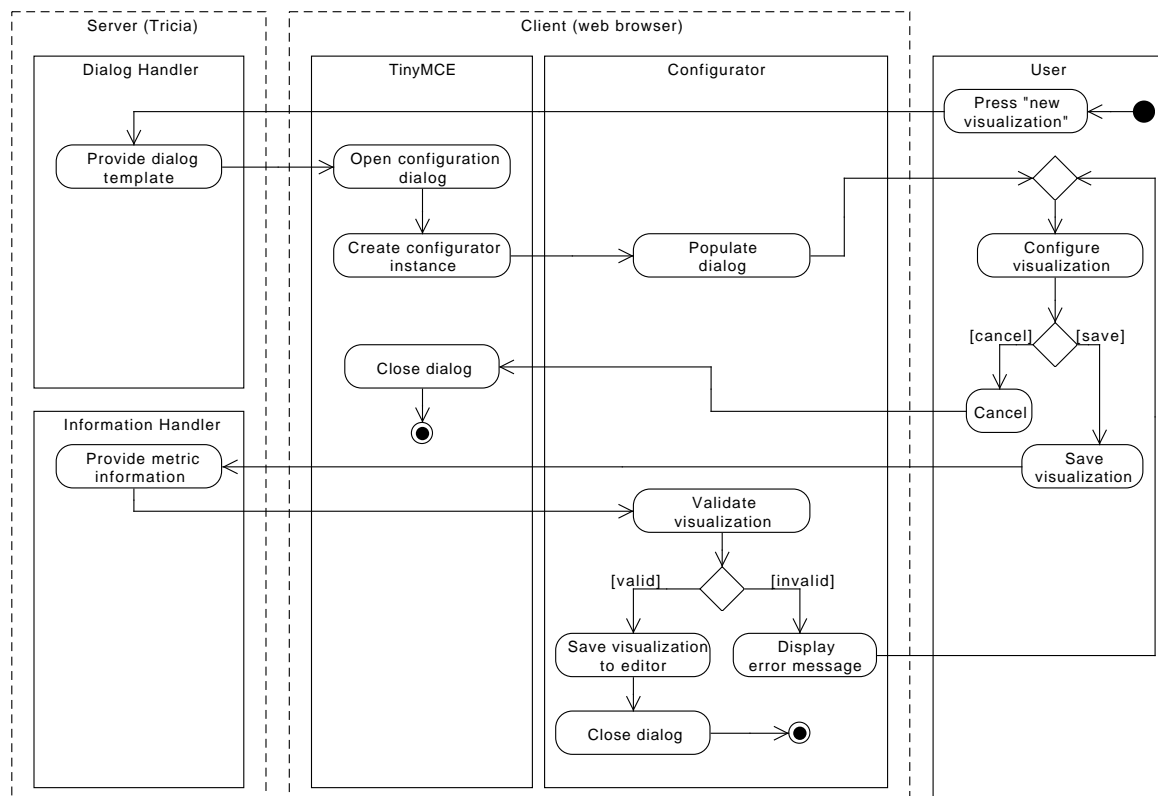


Figure 6.4.: Activity diagram for a new configuration dialog.

The configuration process is as follows:

1. The user presses the corresponding button in the TinyMCE interface to add or edit a visualization.

2. Subsequently the dialog handler is called which provides the empty dialog template.
3. The TinyMCE shows the configuration dialog as overlay and
4. creates an instance of the visualization configurator class.
5. Consequently, the configurator populates the dialog with the available visualization types and with the values of the selected visualization if there is any.
6. The user can now configure the visualization by selecting the desired visualization type, providing the MxL expressions as sources for the data, and perform additional customization as desired as shown in Figure 6.5.
7. Upon pressing the *save*-button, the information handler is called to fetch information about the given MxL expressions.
8. The expression information is subsequently used to validate the visualization, i.e. check whether the expressions exist and fit the criteria detailed in Section 6.2.
9. If the visualization is valid, it is saved to the editor as a Tricia BlockSubstitution and the configuration dialog is closed.
 - At any point, the user can abort the configuration process by pressing the *cancel*-button.
 - If the validation fails, an error message is displayed to inform the user and the visualization is not saved.

After the configuration process is finished, the configuration is embedded in the rich text content of the page as described in Section 6.1.

6.4. Rendering process

As stated above, a visualization configuration is embedded into the rich text content of a Tricia page as BlockSubstitution. On page load, the BlockSubstitution is replaced by the according HTML template, which in turn contains the JavaScript code to calculate the data and render the visualization. Figure 6.6 shows an activity diagram for the rendering

process. Shown are the actions performed by the code of the substitution template, as well as those performed by the calculator component and the server-side handlers.

The process is as follows:


1. The substitution template creates an instance of the visualization calculator class.
 2. The visualization calculator prepares a Highcharts configuration according to the template and the values of the configuration.
 3. Subsequently the information handler is called to fetch information about the given MxL expressions.
 4. The expression information used to validate the visualization, i.e. check whether the expressions still exist and still fit the criteria detailed in Section 6.2. This is done because the expressions might have been changed since the visualization was configured.
 5. If the visualization is valid, the calculator component calls the data handler to calculate the data based on the input from the configuration (date range or categories) and the given MxL expressions.
 6. The result of the calculation is a list of tuples as described in Section 6.2 for every expression. Those data series are incorporated into the Highcharts configuration.
 7. Finally the completed Highcharts configuration is handed to the Highcharts library to render the visualization.
- If the validation fails, an error message is displayed instead of the visualization to inform the user.

All in all, the rendering process leads to the visualizations always being calculated on every page load. This might lead to longer loading times, if the underlying expressions take a long time to be calculated, or if a lot of visualizations are on the same page. To mitigate the impact of those calculation times, caching could be employed, but the caching would be the duty of the underlying MxL implementation and is therefore not in the scope of this thesis.

6.4. Rendering process

Choose visualization type

- Line Chart
- Area Chart
- Column Chart
- Bar Chart
- Pie Chart
- Kiviart Chart
- Bullet Chart**



Choose MxL expressions as data source

Measure

Target

Configure the visualization

Title

Subtitle

X-Axis Type ▼

X-Axis Title

Categories

Y-Axis Type ▼

Y-Axis Title

Show navigator

Show Data Labels

Figure 6.5.: Configuration dialog.

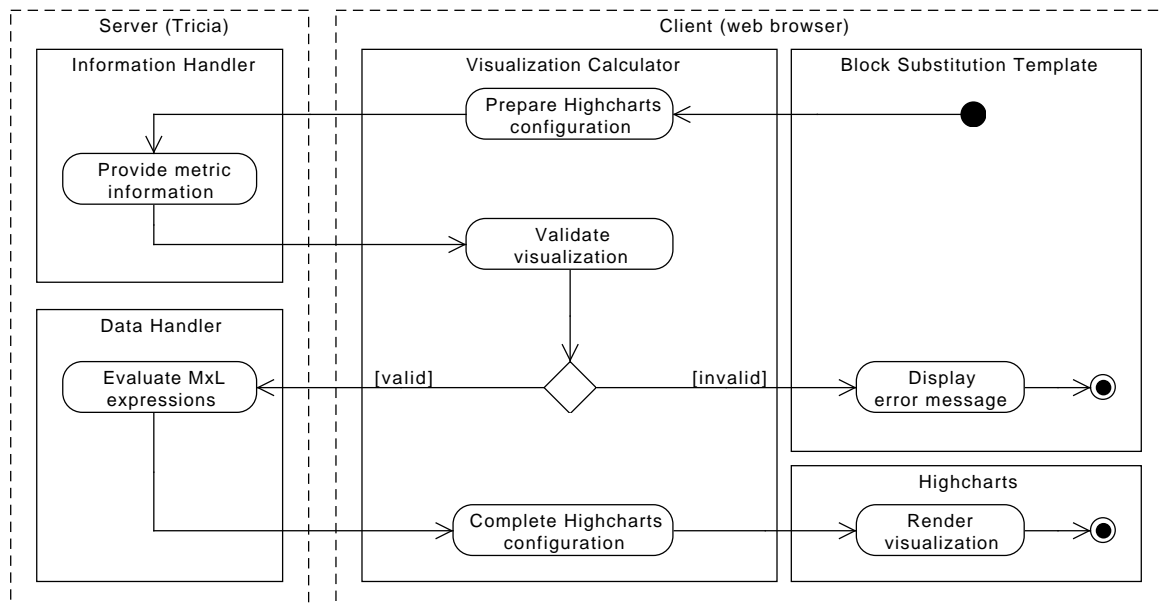


Figure 6.6.: Activity diagram for rendering a visualization.

Part III.

Evaluation & Outlook

7. Evaluation

7.1. Research questions

As stated in Section 1.1, the main goal for this thesis is the development of an integrated software support for the user-defined metric visualization in the domain of EAM in Tricia. To show, whether this goal is fulfilled, this section shows whether all research questions can be answered.

What are recommended visualizations for metrics in literature?

Chapter 3 presented an extensive literature review of relevant literature for the visualization of metrics. As stated in Chapter 4, different authors with different backgrounds recommend different graphical representations for the visualization of metrics, but the following visualizations were commonly found:

- Line charts & area charts (Section 4.1)
- Column charts & bar charts (Section 4.2)
- Pie charts (Section 4.3)
- Kiviat charts (Section 4.4)
- Gauges (Section 4.5)
- Bullet charts (Section 4.5)

This list is not exhaustive, since many variations exist for every chart type (e.g. *stacked* line charts and column charts, *donut-shaped* pie charts, etc.). Nevertheless are those recommended visualizations found in literature, and the question is therefore answered.

What type of visualizations are recommended for metrics in the domain of EAM?

As shown in Table 3.2, our literature search did not unearth literature specific to the visualization of EAM metrics. Nevertheless we identified four different types of metrics in the Metric Catalog and derived a set of recommended visualization types based on our findings in literature. We thereby recommend the following visualizations depending on the result type of the metric and the perspective:

Result type	Perspective	
	Snapshot	Over time
monetary value	bullet chart	line / column chart
multiple values	column chart	line / column chart
number	bullet chart	line / column chart
percentage	bullet chart	line / column chart

As stated in Section 4.6, line charts and column charts are equivalent for the visualization of a metric over time, but have a slight different focus.

What are suitable web-based visualization libraries to support the visualization of metrics in the domain of EAM?

To find a suitable library for the visualization of metrics, we compared four different libraries as shown in Section 5.3. Those libraries were evaluated on the basis of different aspects (e.g. supported visualization types, documentation availability, etc.) and the Highcharts library chosen as a best fit for the thesis. The following table shows the results of the evaluation:

	RGraph	g.Raphaël	NVD3	Highcharts
Line chart	●	●	●	●
Area chart	●	○	●	●
Column chart	●	●	●	●
Bar chart	●	●	●	●
Pie chart	●	●	●	●
Kiviat chart	●	○	○	●
Bullet chart	○	○	●	◐
Dynamic	●	●	●	●
Documentation	●	●	○	●
Interactivity	○	◐	●	●

In the table a ● means that the specific feature is available, a ◐ means, that it is partially available, and a ○ means that it is not available. The explanation for partial availability is in the corresponding section in Section 5.3.

What is a suitable prototypical implementation of a corresponding visualization extension of Tricia?

Based on our findings in the literature review and the visualization library evaluation, we developed a prototypical extension for Tricia to support the visualization of user-defined metrics. This extension was developed following an evolutionary prototyping model according to Davis [1992] based on the requirements defined in Section 3.4. Its iterations are described in detail in Section 5.4.

The prototype supports the configuration of metric visualizations defined as visualization templates (cf. Section 6.1 for the visualization templates and Section 6.3 for the configuration process), which are embedded into the rich-text content of a Tricia page as *BlockSubstitution* (cf. Section 5.1.3). Upon viewing a page with embedded visualizations, the underlying MxL expressions are evaluated and the visualizations displayed (cf. Section 6.4).

Overall, the prototype is able to visualize all metrics of the Metric Catalog, as a snapshot as well as over time, supporting all recommended visualization types.

7.2. Requirements of the prototype

In Section 3.4, we derived four requirements for the prototype, based on our findings in literature and the research questions defined in Section 1.1:

Requirement 1: Support the visualization of all metrics from the Metric Catalog by all types of recommended visualizations.

As shown in Section 5.3, the Highcharts library employed in our prototype supports the visualization of all recommended visualization types. The visualization templates as defined in Section 6.1 have access to every option Highcharts offers and can therefore be used to define a template for every recommended visualization type. Consequently, our prototype contains a pre-defined template for every visualization type presented in Chapter 4 and therefore supports the visualization of all metrics from the Metric Catalog by all types of recommended visualizations.

Requirement 2: The visualizations operate on the results from related MxL expressions.

As stated in Section 5.2, Tricia offers an implementation of MxL 2.0 called TxL 2.0. Our prototype is built as an extension to Tricia and visualizes the calculation results of MxL expressions representing metrics (cf. Section 6.4). The type of MxL expressions is subject in Section 6.2.

Requirement 3: Integration of the visualization prototype in Tricia.

As stated above, the prototype is fully integrated in Tricia. The configuration dialog is integrated into the rich-text editor of Tricia and the visualization configurations are embedded into the rich-text content of a Tricia page as BlockSubstitution (cf. Section 5.1.3).

Requirement 4: The prototype must enable users to define visualizations for their metrics.

Since the visualization configuration dialog is part of Tricia’s rich-text editor, all users can embed visualizations into Tricia pages where they have the appropriate access rights. They are thereby free to choose the desired visualization type and can customize the visualization to their needs (cf. Section 6.3). In addition to that, the *highcharts_expert* attribute of a visualization configuration can be utilized to further customize a visualization instance (cf. Section 6.1). Users therefore can define custom visualizations for all their metrics.

7.3. Critical reflection

As shown above, the thesis fulfilled its goal of developing an integrated software support for the user-defined EAM metric visualization in Tricia. All research questions could be answered satisfactorily and the prototype fulfills all of its requirements. Nevertheless potentials for optimization exist.

User interface

The prototype takes a very practical approach to its user interface, that is, the configuration dialog was built very simplistic without considering user requirements or user interface design best-practices.

Therefore the development of a better user interface is a research opportunity. For example, the configuration dialog has to be analyzed in respect of usability aspects and the design optimized according to the findings (e.g. divided into pages).

Another opportunity is the optimization of the rendering process from a user perspective. Currently, the time it takes to render a visualization is composed of the time needed to calculate all MxL expressions and the time Highcharts needs to actually render the visualization. Therefore, if the calculation of the underlying MxL expressions takes a long time, the visualization doesn’t show up until calculation is finished and the user might assume that something is broken. To alleviate this, for example a loading animation could be added to the rendering process to inform the user that the calculation of the underlying expressions is ongoing.

Furthermore, the configuration dialog is lacking help messages for the user. Currently only the input fields for data sources are validated, but all other input fields (e.g. categories or date ranges) are not. In addition to that, tool tips could be added to guide the user.

Recursive metrics

During the writing of this thesis, our group found a new type of metrics in industry that is currently not addressed. Those metrics are defined *recursively* and encode a part-of relationship. An example for such a metric is *Number of Applications per sub-domain*, whereby every sub-domain can have sub-domains of its own.

The resulting values are best displayed in a *tree*, but graph visualizations were not considered for this thesis due to their absence in literature. Furthermore, Highcharts cannot render trees and therefore another visualization library would be needed.

It might be possible to visualize the results of such a metric as column chart, with the number of applications represented with a column for every sub-domain. However, the part-of relationship of the numbers would be lost and the names of the sub-domains would have to be specified as categories for the visualization. But the recursive nature of the metric cannot ensure that those names are known up front.

Input parameters

Currently, all input parameters for a visualization (e.g. categories, date ranges) are static for the visualization instance, that is, once a user specified a fixed set of categories or a fixed date range, the visualization will be rendered with this input until a user changes the configuration. It is therefore not possible to render a visualization for a relative point in time (e.g. last quarter, last month, etc.) or to automatically update the categories of a visualization if the underlying MxL expression changes. Therefore all affected visualizations have to be edited manually if the current date changes or new possible categories for a metric are introduced.

For example the metric *Incident duration* specifies three severity levels for incidents: low, middle, and high. The metric can be visualized as shown in Figure 4.3 with the

severity levels specified as categories for the corresponding MxL expression. If a fourth severity level is introduced (e.g. *fatal*), all corresponding visualizations have to be updated manually to incorporate the new level.

This behaviour could for example be mitigated by allowing MxL expressions as input for visualizations. Those expressions could either provide a list of categories, or a date range and therefore provide dynamic input for all visualizations.

Visualization reusability

Currently, every visualization configuration is unique and independent from all other visualization configurations. Therefore, if a user wants to incorporate a visualization into multiple pages, the configuration has to be copied to every single page. This is problematic, if the “original” configuration changes, because the changes are not reflected in the “copies”. The user would have to manually keep track of all locations of a visualization and manually propagate the changes to all copied configurations. It is therefore hardly possible to define a visualization once and incorporate it into multiple pages while keeping track of changes.

8. Future research

8.1. Visualization of historical data

As mentioned in Section 6.2, at the writing of this thesis MxL does not support the calculation of historical data. It is therefore not possible to calculate the value of a metric at a point in the past or derive a possible value of a metric at a point in the future.

Proposed solution

Our group is currently developing a new operator as an extension to TxL, which allows the calculation of expressions at an arbitrary point in the past. The operator utilizes the version history of Tricia to calculate the value of an expression based on the state of the hybrid wiki at a point in the past.

8.2. Dynamic configuration dialog

Currently, the customization options of the configuration dialog are hard-coded to the titles, axis types and ranges, and two options for the date navigator and the data labels. Those fields are a sort of least common denominator for all possible visualizations. Therefore, every other visualization instance-specific configuration has to be done through the *highcharts_expert* attribute of the visualization configuration. As stated in Section 6.1, the content of this attribute is not validated and prone to spelling errors.

Since it is not possible to dynamically add input fields to the configuration dialog, therefore every option besides the pre-defined fields common to all visualization types has to be set in the *highcharts_expert* attribute of the visualization configuration. This makes minor changes (e.g. borders, colors, y-axis range and unit, etc.) complicated and prone to

errors, because the users have to translate them to the according part of a Highcharts configuration. The user therefore need intimate knowledge of the Highcharts library, if they want to efficiently customize visualizations.

Proposed solution

To allow an individual configuration dialog for every visualization type, the fields of the configuration dialog have to be encoded in the visualization templates. Therefore a new template attribute would have to be added which contains a definition of every input field of the configuration dialog, together with a mapping to the according Highcharts option. Listing 8.1 contains an example definition for an input field to define a border color and an input field to define a border width for a visualization.

```
1 "dialog": [  
2   {  
3     "label": "Border color",  
4     "type": "color",  
5     "mapped_to": {  
6       "chart": {  
7         "borderColor": "#000000"  
8       }  
9     }  
10  }, {  
11    "label": "Border width",  
12    "type": "number",  
13    "mapped_to": {  
14      "chart": {  
15        "borderWidth": 0  
16      }  
17    }  
18  }  
19 ]
```

Listing 8.1: Exemplary template addition for a dynamic configuration dialog.

This configuration would result in the dialog fields shown in Figure 8.1. The proposed solution will have to be refined to accommodate more complex configuration options, e.g. rule-based color coding or value transformations.



Border Color

Border Width

Figure 8.1.: Resulting dialog fields of Listing 8.1.

8.3. Guided metric selection

Currently the user has to make sure, that the MxL expressions utilized in a visualization exist and conform to the rules detailed in Section 6.2. In addition to that, the input fields for the data sources provide no auto-completion of values and therefore are prone to spelling errors. This makes the data source selection process more tedious, because the user has to switch between different views to choose the correct expressions.

Proposed solution

To aid the user in choosing metrics for a visualization, the input fields need auto-completion support for MxL expression names, automatically filtered to appropriate expressions for the selected visualization type. This auto-completion could be implemented similar to the auto-completion for the search fields in Tricia.

8.4. Graph visualizations

During interviews with practitioners, our group recognized, that there is a need for the visualization of enterprise architectures together with suitable metrics in a graph. That is showing the composition of the EA together with the state of the EA in the form of metrics. Another use case for graph visualizations are recursively defined metrics as stated

in Section 7.3.

Graph visualizations are currently not possible, due to Highcharts not supporting graphs.

Proposed solution

To display graph visualizations an additional rendering engine is needed. Currently Tricia utilizes the Raphaël library to render an UML class diagram of the hybrid types of a wiki. This process would have to be enhanced to add metric information to the resulting diagram. A similar approach with metrics for application landscapes is proposed by Lankes [2008].

Another candidate for graph visualizations is the D3.js¹⁸ library.

8.5. Graphical drill-down on visualizations

In the field of business analytics the *drill-down* is an important concept, referring to the act of going from more aggregated data to less aggregated components of the data. The same concept applies to the visualization of metrics, e.g. if a visualization shows one or multiple metrics for the whole organization and the user wants to view the metrics broken down by different locations.

Proposed solution

The Highcharts library supports a graphical drill-down on all visualization types by supplying the less aggregated data series for every value in a visualization. To utilize this option, the user would either have to supply additional MxL expressions for the calculation of the less aggregated data series, or the implementation has to automatically calculate the data from the composition of the MxL expression itself.

Both approaches have several disadvantages:

¹⁸<http://d3js.org>

Additional expressions In this case, the user has to manually define the appropriate expressions for every step of the drill-down and somehow supply them during the configuration process of a visualization. Therefore the user would have to know how the drill-down works and make sure, that the correct expressions are used to calculate the drill-down values, which might introduce additional sources of error.

Automatic calculation In this case, the expressions necessary to calculate the drill-down values are automatically derived from the expression representing the metric. At the time of writing this thesis, this is not possible in the current implementation of MxL in Tricia. In addition to that, the application would have to decide along which axis the drill-down is made (e.g. drill-down to locations or drill-down to business applications).

Further research is needed to determine the best strategy to realize a graphical drill-down on the metric visualizations, since there seems to be no obvious solution to the problem.

Bibliography

- Frederik Ahlemann, Eric Stettiner, Marcus Messerschmidt, and Christine Legner. *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*. Springer, 2012.
- Herbert Antoine. Die graphische Darstellung von Betriebskennzahlen. In *Kennzahlen, Richtzahlen, Planungszahlen*, pages 151–156. Springer, 1956.
- Heinz-Josef Botthof, Franz Hölzl, and Nadja Raslan. *Wie Zahlen wirken: betriebliche Kennzahlen vorteilhaft darstellen*. Haufe-Lexware, 2008.
- Sabine Buckl, Florian Matthes, Christian Neubert, and Christian M Schweda. A wiki-based approach to enterprise architecture documentation and analysis. In *ECIS*, pages 1476–1487, 2009.
- Sabine Buckl, Thomas Dierl, Florian Matthes, and Christian M Schweda. Building Blocks for Enterprise Architecture Management Solutions. In *Practice-Driven Research on Enterprise Transformation*, pages 17–46. Springer, 2010.
- Sabine M. Buckl. *Developing Organization-Specific Enterprise Architecture Management Functions Using a Method Base*. PhD thesis, Technische Universität München, München, 2011.
- Tobias Burger. *Management Cockpit: Unterstützung von Kennzahlen durch eine effiziente Visualisierung*. LFI, Abt. für Handwerkswissenschaften, 2006.
- Thomas Büchner, Florian Matthes, and Christian Neubert. Data model driven implementation of web cooperation systems with tricia. In *Objects and Databases*, pages 70–84. Springer, 2010.

- Evellin Cristine Souza Cardoso. Challenges in performance analysis in enterprise architectures. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International*, pages 327–336. IEEE, 2013.
- William S Cleveland. *The elements of graphing data*. Wadsworth advanced books and software Monterey, CA, 1985.
- William S Cleveland and Robert McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.
- William S Cleveland and Robert McGill. Graphical perception and graphical methods for analyzing scientific data. *Science*, 229(4716):828–833, 1985.
- Alan M Davis. Operational prototyping: A new development approach. *Software, IEEE*, 9(5):70–78, 1992.
- Paul W Farris, Neil T Bendle, Phillip E Pfeifer, and David J Reibstein. *Marketing metrics: The definitive guide to measuring marketing performance*. Pearson Education, 2010.
- Stephen Few. Keep Radar Graphs Below the Radar - Far Below. http://www.perceptualedge.com/articles/dmreview/radar_graphs.pdf, 2005. [Online; accessed 2014-04-08].
- Stephen Few. *Information dashboard design*. O’Reilly, 2006.
- Stephen Few. Save the Pies for Dessert. <http://www.perceptualedge.com/articles/08-21-07.pdf>, 2007. [Online; accessed 2014-04-08].
- Stephen Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press, USA, 1st edition, 2009. ISBN 0970601980, 9780970601988.
- Fiorenzo Franceschini, Maurizio Galetto, and Domenico Maisano. *Management by Measurement: Designing Key Indicators And Performance Measurement Systems*. Springer, 2007.
- Shirley Gregor and Alan R Hevner. Positioning and presenting design science research for maximum impact. *MIS Quarterly*, 37(2), 2013.

- Inge Hanschke. *Strategic IT Management: A Toolkit for Enterprise Architecture Management*. Springer, 2009.
- Inge Hanschke. *Strategisches Management der IT-Landschaft*. Hanser, München, 2010.
- Inge Hanschke. *Enterprise architecture management-einfach und effektiv: ein praktischer Leitfaden für die Einführung von EAM*. Carl Hanser Verlag GmbH Co KG, 2013.
- Inge Hanschke, Gunnar Giesinger, and Daniel Goetze. *Business-Analyse einfach und effektiv: Geschäftsanforderungen verstehen und in IT-Lösungen umsetzen*. Carl Hanser Verlag GmbH Co KG, 2013.
- ITGI. Cobit 4.1. Technical report, IT Governance Institute. Rolling Meadows, IL, USA, 2009.
- W Michael Kelley and Robert AD Donnelly Jr. *The Humongous Book of Statistics Problems: Translated for People Who Don't Speak Math*. Penguin, 2009.
- Harold R Kerzner. *Project Management Metrics, Kpis, and Dashboards: A Guide to Measuring and Monitoring Project Performance*. John Wiley & Sons, 2013.
- Rolf Knoll and Christopher Schulz. *Enterprise Architecture Tool Survey 2013*. Syracom AG, 2013.
- Josef K Lankes. *Metrics for Application Landscapes: Status Quo, Development, and a Case Study*. PhD thesis, Technische Universität München, 2008.
- Yair Levy and Timothy J Ellis. A systems approach to conduct an effective literature review in support of information systems research. *Informing Science*, 9, 2006.
- Carsten Lucke, Sascha Krell, and Ulrike Lechner. Critical issues in enterprise architecting—a literature review. *16th Americas Conference on Information Systems*, 2010.
- Florian Matthes and Christian Neubert. Wiki4eam: Using hybrid wikis for enterprise architecture management. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 226–226. ACM, 2011.
- Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M Schweda. *Enterprise Architecture Management Tool Survey 2008*. Technische Universität München, 2008.

- Florian Matthes, Ivan Monahov, Alexander Schneider, and Christopher Schulz. Metric Catalog. Technical report, Technische Universität München, 2012a.
- Florian Matthes, Ivan Monahov, Alexander Schneider, and Christopher Schulz. A template-based design method to define organization-specific kpis for the domain of enterprise architecture management. In *DASMA Software Metrik Kongress*, 2012b.
- Florian Matthes, Ivan Monahov, Alexander Schneider, and Christopher Schulz. Towards a unified and configurable structure for ea management kpis, 2012c.
- Microsoft. LINQ (Language-Integrated Query). <http://msdn.microsoft.com/en-us/library/vstudio/bb397926.aspx>, 2013. [Online; accessed 2014-04-17].
- Ivan Monahov, Thomas Reschenhofer, and Florian Matthes. Design and prototypical implementation of a language empowering business users to define key performance indicators for enterprise architecture management. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International*, pages 337–346. IEEE, 2013.
- Tamara Munzner. Process and pitfalls in writing information visualization research papers. In *Information visualization*, pages 134–153. Springer, 2008.
- Christian Neubert. *Facilitating Emergent and Adaptive Information Structures in Enterprise 2.0 Platforms*. PhD thesis, Technische Universität München, München, 2012.
- Object Management Group (OMG). Object Constraint Language. <http://www.omg.org/spec/OCL/2.3.1/>, 2012. [Online; accessed 2014-04-17].
- Office of Government Commerce (OGC). *ITIL – Service Delivery*. IT Infrastructure Library (ITIL). The Stationery Office. Norwich, UK., 2000.
- David Parmenter. *Key Performance Indicators: Developing, Implementing, and Using Winning KPIs*. John Wiley & Sons, 2010.
- Nils H Rasmussen, Manish Bansal, and Claire Y Chen. *Business Dashboards: A Visual Catalog for Design and Deployment*. John Wiley & Sons, 2009.

- Joachim Rawolle, Ralf Grosser, and Manfred Stojka. It-basierte prozesssteuerungs-geschäftsprozessmonitoring am beispiel der versicherungsindustrie. *Controlling & Management*, 52:76–83, 2008.
- Thomas Reschenhofer. Design and prototypical implementation of a model-based structure for the definition and calculation of enterprise architecture key performance indicators. Master’s thesis, Technische Universität München, 2013.
- Christian M. Schweda. *Development of Organization-Specific Enterprise Architecture Modeling Languages Using Building Blocks*. PhD thesis, Technische Universität München, München, 2011.
- Julie Short. Magic quadrant for enterprise architecture tools, 2013.
- Matthias Stutz. *Kennzahlen für Unternehmensarchitekturen: Entwicklung einer Methode zum Aufbau eines Kennzahlensystems für die wertorientierte Steuerung der Veränderung von Unternehmensarchitekturen*. PhD thesis, Universität St. Gallen, 2009.
- Erdisa Subashi. Establishing kpi systems for enterprise architectures: Risks and countermeasures. Master’s thesis, Technische Universität München, 2013.
- Rebecca Tiarks. Software-monitoring mit hilfe eines dashboards. Master’s thesis, Universität Bremen, 2008.
- Edward R Tufte. Envisioning information. *Optometry & Vision Science*, 68(4):322–324, 1991.
- Edward R Tufte. *The Visual Display of Quantitative Information*, volume 2. Graphics Press Cheshire, CT, 2001.
- Johan E Wallin. Interactive balanced scorecard visualization. Master’s thesis, Chalmers University of Technology, 2009.
- Colin Ware. *Information Visualization: Perception for Design*. Elsevier, 2012.
- Horst Wildemann. Visualisierung als controlling-instrument. *Neue Organisationsformen im Unternehmen. Ein Handbuch für das moderne Management*, Berlin ua, pages 946–952, 1996.

André Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. PhD thesis, Technische Universität München, 2007.

World Wide Web Consortium (W3C). W3C Candidate Recommendation. <http://www.w3.org/TR/html5/scripting-1.html#the-canvas-element>, 2014. [Online; accessed 2014-04-03].

A. Appendix

```
1 {
2   "properties": {
3     "name": "Line Chart",
4     "x_axis_types": ["datetime"],
5     "y_axis_types": ["linear", "logarithmic"],
6     "no_of_metrics": "multiple",
7     "title_editable": true,
8     "subtitle_editable": true,
9     "x_axis_type_editable": false,
10    "x_axis_title_editable": true,
11    "y_axis_type_editable": true,
12    "y_axis_title_editable": true,
13    "data_labels_editable": true,
14    "navigator_editable": true,
15    "date_range_editable": true,
16    "date_interval_editable": true,
17    "categories_editable": true,
18    "default_categories": [],
19    "default_date_range": [],
20    "default_date_interval": "monthly",
21    "date_intervals": ["daily", "weekly", "monthly", "quarterly", "annually"],
22    "metric_series": [],
23    "metric_series_addition": {
24      }
25  },
26  "configuration": {
27    "categories": [],
28    "date_range": [],
29    "date_interval": "",
30    "title": "",
31    "subtitle": "",
32    "x_axis_type": "",
33    "x_axis_title": "",
34    "y_axis_type": "",
35    "y_axis_title": "",
36    "navigator": true,
37    "data_label": false,
38    "series": [],
39    "container": ""
40  },
41  "highcharts_config": {
42    "chart": {
43      "type": "line",
```

```
44     "renderTo": ""
45   },
46   "title": {
47     "text": ""
48   },
49   "subtitle": {
50     "text": ""
51   },
52   "xAxis": {
53     "type": "datetime",
54     "title": {
55       "text": ""
56     },
57     "lineWidth": 1
58   },
59   "yAxis": {
60     "type": "linear",
61     "title": {
62       "text": ""
63     },
64     "lineWidth": 1
65   },
66   "navigator": {
67     "enabled": true
68   },
69   "plotOptions": {
70     "line": {
71       "dataLabels": {
72         "enabled": false
73       }
74     }
75   },
76   "tooltip": {
77     "shared": true
78   },
79   "series": []
80 },
81 "highcharts_expert": {
82 },
83 }
```

Listing A.1: Complete line chart template.

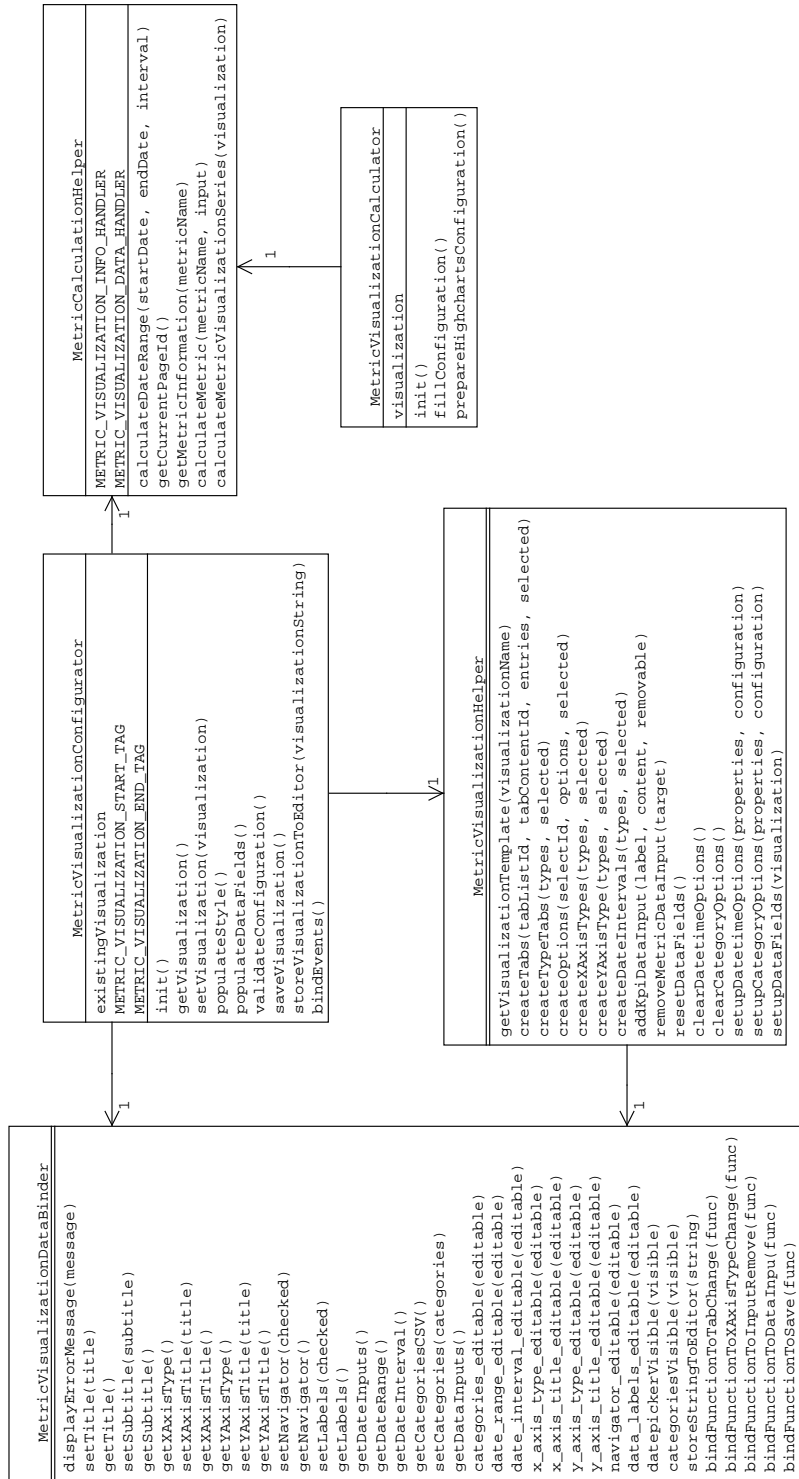


Figure A.1.: Full class diagram of the visualization configurator.