5. Related Work

Be less curious about people and more curious about ideas.

(Marie Curie)

Inhaltsangabe

Der erste Abschnitt dieses Kapitels beschäftigt sich mit der Funktionsweise der Metrikmanagement-Datenbank. Es wird darauf eingegangen, in welcher Form Metriken erfasst werden und nach welchem Schema das Konzept der Wiederverwendung bei der Erstellung neuer Metrik-Spezifikationen Anwendung findet.

Anschließend wird ein Konzept zum systematischen werkzeugunterstützten Anpassen von Metriken vorgestellt und bewertet. Insbesondere soll die Methodik des Metrik-Frameworks herausgestellt werden.

Im letzten Teil dieses Kapitels wird die Arbeitsweise der Dashboard-Webplattform *Geckoboard* untersucht. Das Hauptaugenmerk liegt dabei auf der Funktionsweise der Widgets und des Widget-Editors mit dem sich eigene Widgets mit benutzerdefinierter Variabilität erstellen lassen.

5.1. Metrikmanagement-Datenbank

Die Metrikmanagement-Datenbank ist ein Teil der MeDIC-Werkzeugsammlung und dient der Dokumentation und Verwaltung von Metriken ([12]). Das System ermöglicht die textuelle Definition von Metriken anhand bestimmter Attribute wie *Name*, *Skala*, *Wertebereich*, *Einheit*, *Messgegenstand* oder *Beschreibung* (siehe Abbildung 5.1). Zusätzlich erlaubt die Oberfläche, ähnlich wie in MeDIC-Dashboard, eine Verknüpfung von Metrik und Informationsbedürfnissen. Eine Metrik kann entweder vom Typ *Basismetrik* sein oder vom Typ *Abgeleitete Metrik*. Letzteres bedeutet, dass zusätzlich Assoziationen zu den passenden Ausgangsmetriken erfasst werden.

Abbildung 5.1.: Metrikmanagement-Datenbank Bildschirmfoto (aus [12])

Des Weiteren wird ein systematischer Vorschlag- und Bewertungsprozess für Metriken gemäß CMMI unterstützt. Eine Versionierung erlaubt die iterative Entwicklung neuer Metriken, optional als Variation von vorgegebenen Standardmetriken. Die Standardmetriken dienen dabei als Vorlage und ermöglichen damit die Wiederverwendung von Wissen und Erfahrung.

5.2. Systematisches werkzeugunterstütztes Anpassen von Metriken

Die Diplomarbeit von Ricardo Tavizón [53] beschäftigt sich mit der Konzeption eines Rahmenwerks zum werkzeugunterstützten Anpassen von Metriken. Die Nutzung von Metriken in einem Unternehmen spielt eine wichtige Rolle zur Beurteilung von Entwicklungsphasen und [31]den eingesetzten Technologien.[1] Der Prozess des Metrik-Managements bezeichnet die Aufgabe Metriken zu entwickeln und die Adaption von im Unternehmen genutzten Metriken. Zu diesem Zweck wurden verschiedene Ansätze betrachtet, die den systematischen Entwurf von Metriken erlauben.

[1] Für weitere Details zu Metriken siehe Ludewig und Lichter [31] oder Tavizón [53]

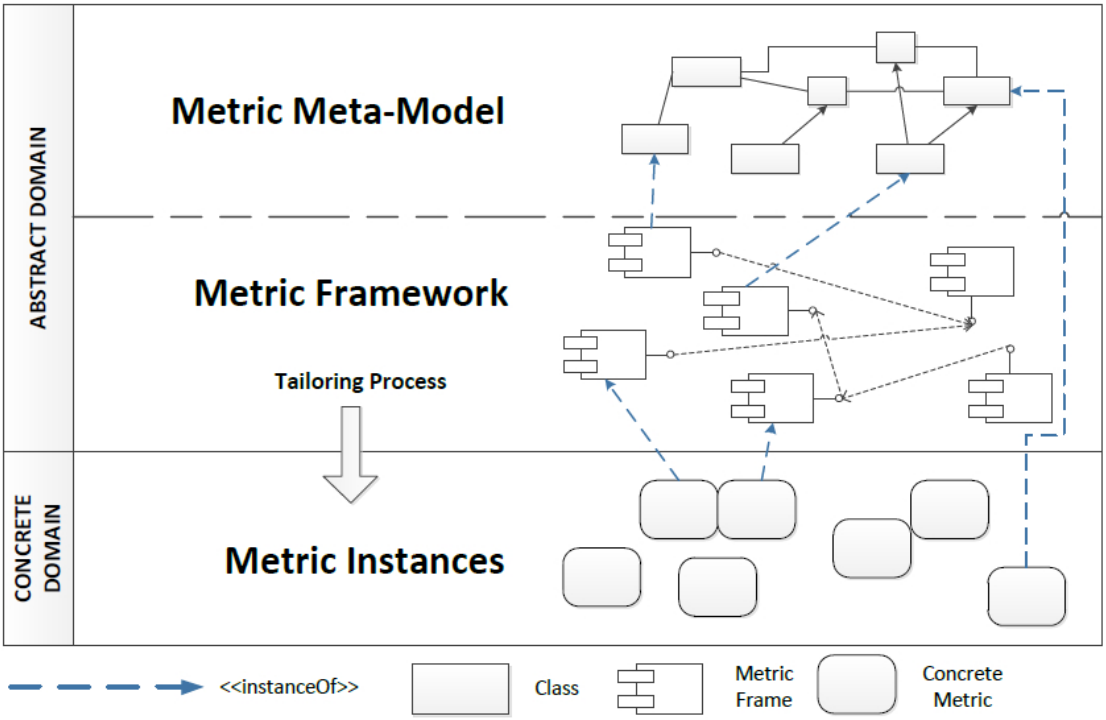

Abbildung 5.2.: Metrik-Framework (aus [53])

Abbildung 5.2 zeigt eine makroskopische Ansicht auf das entwickelte Konzept, welches hierarchisch in drei Schichten unterteilt ist. Die *Abstract Domain* umfasst das *Metrik-Meta-Modell* und das *Metrik-Framework*.

Metric Meta-Model Basierend auf den Begriffen und Konzepten des Messinformationsmodells nach ISO15939, enthält diese Schicht Regeln und ein Vokabular für die Konstruktion von Metriken.

Metrik-Framework Das Metrik-Framework bietet die nötige Unterstützung zur Repräsentation von metrik-bezogenem Wissen in einem Unternehmen. Es besteht aus einer Menge von vorgefertigten *Metrik-Frames*, welchen für die Wiederverwendung gedacht sind. Ein Metrik-Frame enthält eine Metrik-Spezifikation und ein Variabilitätsmodell zur Repräsentation der veränderlichen Attribute der Metrik. Das Variabilitätsmodell ist an das orthogonale Variabilitätsmodell aus Abschnitt 4.3 angelehnt. Die Metrik-Frames werden anhand von zwei Merkmalen kategorisiert: Erstens für welchen Projekttyp sie bestimmt sind und zweitens, ob es sich um eine Basismetrik oder eine abgeleitete Metrik handelt. Eine Metrik kann demnach auf den Messwerten anderer Metrik-Frames aufbauen. In Abbildung 5.2 ist das grafisch durch eine Analogie zu Komponenten und ihren Wechselbeziehungen angedeutet.

Abbildung 5.3.: Geckoboard Beispiel-Dashboard

Die Schicht der *Metrik-Instanzen* bildet die *Concrete Domain* und enthält projektspezifische Instanzen der Metrik-Frames. Eine Metrik-Instanz ist eine abgeleitete Instanz eines Metrik-Frames. Dieser Prozess wird als *Tailoring* bezeichnet. Beim Tailoring werden die Varianten des Metric-Frames ausgewählt und man erhält eine konkrete Metrik, welche einen Teil der Informationsbedürfnisse des entsprechenden Projekts erfüllen.

Zusätzlich zu diesem abstrakten Konzept wurde ein Papierprototyp entwickelt, der die Erstellung von Metrik-Frames, ihren Wechselbeziehungen und das Tailoring der Metrik-Instanzen unterstützt.

Die Evaluierung des Prototyps hat jedoch ergeben, dass der Nutzer zur Bedienung des Werkzeugs ein tiefes Verständnis für das Metrik-Framework und die Verwaltung der Variabilität haben muss. Zudem erwies sich die Dialoggestaltung als unübersichtlich, überladen und führte zu einer relativ komplizierten Bedienung.

5.3. Geckoboard

Geckoboard [20] ist ein Mehrzweck-Dashboard zur Visualisierung von Messdaten mit der Unterstützung für eine Vielzahl an unterschiedlichen Datenquellen (Abb. 5.3). Laut Herstellerangaben wurde die grafische Darstellung des Dashboards bewusst Einfachheit

gehalten um den Fokus auf die gemessenen Werte zu lenken. Die Anzahl der direkt unterstützten Datenquellen beläuft sich aktuell auf 106[2]. Dabei erfolgt die Anbindung über die jeweilige Webservice-Schnittstelle des Datenlieferanten. Das Spektrum der Datenlieferanten ist sehr breit gefächert. Es reicht von Systemen für Web Analytics, Customer Relationship Management und Social Media bis hin zu Systemen für die kontinuierliche Integration, Projektmanagement, und Test-Frameworks.

Einige konkrete Beispiele für Systeme, die in die letzten drei genannten Bereiche fallen sind *Github*, *Jira*, *Jenkins*, *Sprintly*, *Basecamp* oder *Sismo*.

Obwohl die integrierten Systeme möglicherweise eine autonome Dashboard-Funktionalität besitzen, ermöglicht Geckoboard Daten aus verschiedenen Quellen in einem Dashboard zu vereinen.

Für die Visualisierung von Metriken in Form von Diagrammen oder speziell formatiertem Text sind sogenannte Widgets zuständig. Die Anordnung der Widgets erfolgt entlang eines unsichtbaren Gitters. Man kann sie mit der Maus verschieben und an gewünschter Position automatisch einrasten lassen.

Geckoboard liefert für jede unterstützte Datenquelle eine Reihe vorgefertigter Standardwidgets mit, die mit einer definierten Variabilität ausgestattet sind. Das heißt sie stellen die Möglichkeit der benutzerdefinierten Konfiguration des entsprechenden Widgets bereit. Die Einstellungsmöglichkeiten können zum Einen die Datenquelle direkt betreffen und haben somit Einfluss auf die darzustellenden Messdaten. Zum Anderen erlauben sie die Anpassung der visuellen Repräsentation der Messdaten.

Für die Datenquelle *Google Analytics*[3] gibt es beispielsweise ein Widget zur Darstellung von Besucherstatistiken. Tabelle 5.1 zeigt eine Auswahl der verfügbaren veränderlichen Widget-Attribute mit den dazugehörigen Varianten (per Komma getrennt).

Attribut	**Varianten**
Label (optional)	*Freitext*
Widget size	1x1, 2x2
Accounts	Website 1, Website 2, etc.
Metrics	Visits, Pageviews, Pages/Visit, Bounce Rate, Avg. Time on Site, % New Visits, Unique visitors
Filters (optional)	*Freitext*[4]
Period	Today, Yesterday, Past 7 Days, Past 28 Days, Past 30 Days, Past 60 Days, Past 90 Days, Past 180 Days, Past 365 Days
Secondary Output	% Change this period, Sparkline

Tabelle 5.1.: Auswahl der Attribute eines Geckoboard-Widgets für Google Analytics

[2]Stand 31.05.2013; vergleiche http://www.geckoboard.com/widget-directory/
[3]Google Analytics ist ein Dienst zur detaillierten Analyse von Website-Traffic.
[4]http://code.google.com/apis/analytics/docs/gdata/v2/gdataReferenceDataFeed.html#filters

Ein interessanter Aspekt bei Geckoboard ist, dass der Zustand eines Dashboards, mit all seinen Messwerten, beim Client im Browser zwischengespeichert wird. Dies wird mit Hilfe der HTML5 Web Storage API [40] ermöglicht, welche von allen wichtigen Browsern nativ unterstützt wird[5]. Auf diese Weise können zuletzt abgerufene Messdaten weiterhin angezeigt werden, auch wenn die jeweiligen Service-APIs vorübergehend nicht erreichbar sind. Die Messwerte werden über die Webservice-Schnittstellen der jeweiligen Datenlieferanten eingesammelt. Manchmal wird dazu ein Umweg über die Geckoboard-Data-API genommen, um Probleme der Same-Origin-Policy zu vermeiden. Die Data-API agiert dabei offenbar lediglich als Vermittler. Es bleibt festzuhalten, dass Geckoboard auf den ersten Blick einen großen Teil der Geschäftslogik in die Client-Schicht auslagert und ein relativ schlichtes Server-Backend besitzt. Diese Erkenntnisse basieren auf persönlichen Nachforschungen und Tests und sind nicht von offizieller Seite bestätigt.

5.3.1. Widget Editor

Geckoboard bietet Entwicklern die Möglichkeit eigene Widgets mit zugehöriger Variabilität zu erstellen und somit auch neue Datenlieferanten zu integrieren. Die Werbservice-Schnittstelle muss dazu in der Regel nicht verändert werden. Der gesamte Widget-Quellcode basiert auf JavaScript und wird im Client ausgeführt.

Wie Tabelle 5.2 zu entnehmen ist, besteht ein benutzerdefiniertes Geckoboard-Widget aus vier Dateien.

In der Konfigurationsdatei `widget.json` gibt es einen Abschnitt namens *Configuration Screen*, der es ermöglicht die veränderlichen Attribute des Widgets zu definieren. Configuration Screen steht für den Einstellungsdialog des Widgets. In diesem Abschnitt lässt sich eine Liste von Eingabefeldern *fields* definieren, welche dem Benutzer präsentiert werden sollen. Unterstützt werden drei Arten von Feldern:

Textfeld (*input*) Ein standardmäßiges Textfeld für die Angabe von Freitext.

Dropdown-Listenfeld (*select*) Dient der Einfachauswahl der Elemente einer Liste von vorgegebenen Varianten.

Gruppe von Checkboxen (*checkbox*) Möglichkeit zur Mehrfachauswahl der Elemente einer Liste von vorgegeben Varianten.

Alle drei Arten haben folgende Pflichtangaben gemein:

- **name** Der Name des Attributs, dessen Wert von `widget.js` aus adressiert werden kann.
- **label** Die Beschriftung des Eingabefelds, welche im Einstellungsdialog erscheint.

[5] sogar vom Internet Explorer ab Version 8 aufwärts

- **type** Die Art des Eingabefelds: `input`, `select` oder `checkbox`
- **help** Der Kurzinfo-Hilfetext, der im Einstellungsdialog an entsprechender Stelle erscheint.

Datei	Beschreibung	benötigt
widget.json	Die Konfiguration des Widgets erfolgt im JSON-Format. Diese umfasst die Definition der Datenquellen, den Diagrammtyp, erlaubte Widget-Größen, Informationen zum Entwickler und viele weitere Meta-Informationen. Unterstützte Rückgabeformate der Datenquellen sind XML, JSON und CVS. Außerdem wird hier die Variabilität in Form von veränderlichen Attributen definiert.	ja
widget.js	Diese Datei enthält ein ausführbares Skript, welches mit den Webservice-Schnittstellen der Datenquellen kommuniziert, die relevanten Messinformationen abruft und für die Weiterverarbeitung durch Geckoboard vorbereitet.	ja
config.js	Dieses Skript sorgt optional für das Nachladen bestimmter Varianten im Einstellungsdialog des Widgets.	nein
highcharts.js	Falls die zehn vordefinierten Diagrammtypen nicht ausreichen, lässt sich in dieser Datei mit Hilfe der Highcharts-Bibliothek[6] ein angepasstes Diagramm erstellen.	nein

Tabelle 5.2.: Dateien eines benutzerdefinierten Geckoboard-Widgets

Das Dropdown-Listenfeld und die Gruppe von Checkboxen besitzt ein weiteres Attribut namens **options**, welches eine Liste von vordefinierten Varianten enthält. Diese Liste von Varianten kann auch dynamisch in `config.js` aufgefüllt werden. Dieses Skript wird beim Öffnen des Einstellungsdialogs ausgeführt und kann zur Ermittlung der dynamischen Varianten mit einem Webservice kommunizieren.

Listing 5.1 zeigt eine Beispiel-Konfiguration eines Einstellungsdialogs mit einem Textfeld und einer Gruppe von Checkboxen zur Mehrfachauswahl.

```
1 "configScreen": {
2   "fields": [
3     {
4       "name" : "term",
5       "label" : "Search Term",
6       "type" : "input",
```

[6] Highcharts ist eine JavaScript-Bibliothek zur Generierung von Diagrammen

```
7         "help" : "Input a general search term here"
8       }
9     ],
10    {
11      "name":"status",
12      "help":"Choose the ticket types you wish to see metrics for.",
13      "type":"checkbox",
14      "label":"Choose Ticket Status",
15      "options":[
16        {
17          "name":"New",
18          "value":"new"
19        },
20        {
21          "name":"Open",
22          "value":"open"
23        },
24        {
25          "name":"Closed",
26          "value":"closed"
27        }
28      ]
29    }
30  }
```

Quelltext 5.1: Auszug aus widget.json (aus [20])

Das Eingabefeld `term` ist ein einfaches Textfeld, wohingegen das Feld `status` eine Gruppe von Checkboxen mit den Varianten `new`, `open` und `closed` repräsentiert. Wenn der Benutzer im Dashboard die Aktion *Edit Widget* ausführt, generiert Geckoboard aus dieser Beschreibung einen Einstellungsdialog mit den entsprechenden Eingabefeldern. Falls vorhanden, wird zusätzlich das Skript `config.js` ausgeführt um bestimmte Varianten dynamisch zu generieren.

Nun kann der Benutzer beliebige Änderungen mit Hilfe der Eingabefelder vornehmen und diese anschließend mit dem Betätigen der Schaltfläche *Update Widget* übernehmen. Im Anschluss daran führt Geckoboard das Skript `widget.js` aus, welches die Werte der ausgewählten Varianten der Eingabefelder erkennt und eine oder mehrere Anfragen an die festgelegten Webservice-Schnittstellen der Datenquellen mit den entsprechenden Parametern ausführt.

Dieses Konzept erlaubt die Anbindung neuer Datenquellen auf relativ simple Art und Weise. Nicht berücksichtigt werden jedoch gängige Datentypen wie `Date`, `Float` oder `Integer`. Außerdem gibt es keine Möglichkeit Beschränkungen für einzelne Varianten festzulegen. Wie zum Beispiel: Variante A erfordert Variante B oder Variante C schließt Variante D aus (vgl. Kapitel 4). Positiv ist, dass der Webservice der zu integrierenden Datenquelle nicht verändert werden muss, falls das Ausgabeformat eines der drei Formate XML, JSON oder CSV aufweist. Die Konvertierung und Weiterverarbeitung der Daten

und die Generierung grafischen Darstellung in Form von Diagrammen erfolgt ebenfalls in `widget.js`. Zur Generierung der Diagramme stellt Geckoboard Javascript-Komponenten zur Verfügung.

Im Vergleich zu MeDIC benutzt Geckoboard ein Architekturmodell, dass sehr stark auf die Client-Schicht ausgerichtet ist. Ein großer Teil der Geschäftslogik läuft in Form von Javascript im Browser des Clients. Die Kommunikation mit dem Server-Backend läuft hauptsächlich über einen Webservice (Data-API). MeDIC besitzt im Gegensatz dazu eine klassische JavaEE-Architektur, bei der die Geschäftslogik in der Business-Schicht in Form von Enterprise JavaBeans (EJB) implementiert ist. Jedes definierte Ereignis im Client löst serverseitig den JSF-Lebenszyklus (siehe Abschnitt 2.1.3) aus, welcher wiederum die Funktionalität von bestimmten EJBs in Anspruch nehmen kann.