UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

VERITATE SOLA NOBIS IMPONETUR VIRILISTOGA. 1948

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

LICENCIATURA EN CIENCIA DE LA COMPUTACIÓN

Trabajo de Diploma

"Diseño de una herramienta integrada para la evaluación de la calidad de metadatos bibliográficos"

Autor: Angel Manuel Palmero Albelo

Tutores: MSc. Lisandra Díaz de la Paz
MSc. Juan Luís García Mendoza

Santa Clara, 2017

CON SU ENTRAÑABLE TRANSPARENCIA

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de pregrado en la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma total como parcial y que además no podrá ser presentado en eventos ni publicado sin previa autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor Firma del tutor

_____

Firma del jefe de laboratorio

Dedicatoria

A mi abuelo Osvaldo y mi abuela Erótida, por su inmenso amor y sabiduría.

A mis padres Nelis Yumara y Angel Manuel por hacer posible mi sueño de ser universitario.

A mi hermano Manuel Alejandro que siempre está atento y dispuesto para mí.

Y a mi fiel amiga y compañera, Liset, por apoyarme, por ser mi alma gemela y por estar siempre a mi lado sin dudar.

Agradecimientos

A mis padres, Nelis Yumara y Angel Manuel, por no claudicar en su intento de hacerme mejor persona.

A mi abuelo, Osvaldo, quien es y será siempre más que un padre para mí, mi ejemplo de vida.

A mi abuela Erótida, que, aunque ya no está con nosotros, siempre será mi principal guía, por su amor, apoyo, dedicación y preocupación constante.

A mi hermano Manuel Alejando por su entrega infinita y apoyo incondicional durante esta etapa de mi vida.

A Liset y su familia quienes me dieron tantos abrazos de consuelo, palabras de apoyo y cada momento de comprensión.

A mis hermanas, mi madrastra, mi padrastro, mis tíos y mis primos por su preocupación constante.

A mis tutores Lisandra y Juan Luis, quienes siempre confiaron en mí. Gracias por darme la fortaleza y confianza, gracias por enseñarme que el miedo es pasajero y el conocimiento perdurable.

A mis amigos quienes siempre estuvieron a mi lado y me sobrellevaron todos estos años y me apoyaron en todos los procesos de diseño y realización de esta investigación.

Y a todos aquellos que, aunque no alcancen las palabras ni el espacio para mencionarlos saben y están conscientes de que mi agradecimiento les llegará de forma directa y de corazón. A esos que por problemas de memoria no menciono, mis disculpas y gracias por existir.

A todos, los amo y espero que nunca desaparezcan de mi vida, pues forman parte de la etapa más importante que cualquier persona puede imaginar.

Muchas gracias.

RESUMEN

En trabajos de diploma desarrollados en el período comprendido entre el año 2015 y 2017 se implementaron herramientas aisladas que brindan soluciones a algunos problemas de calidad de los metadatos bibliográficos catalogados en bibliotecas y centros de documentación del país. Sin embargo, estas por sí solas no brindan una visión íntegra de todo el ciclo de vida de la calidad de los datos. Por consiguiente, el objetivo principal del presente trabajo es proponer el diseño de una herramienta integrada, extensible y escalable, que permita la evaluación de la calidad de los metadatos bibliográficos, haciendo uso de patrones de diseño y del patrón arquitectónico Modelo Vista Controlador. Para ello se realiza un análisis crítico de varias metodologías de calidad de los datos, con el fin de determinar las fases que se ajustan a los metadatos bibliográficos y se realiza un análisis crítico del diseño de las herramientas implementadas hasta el momento de manera independiente. Como resultado, se propone el diseño de una arquitectura y de la interfaz visual de una herramienta integrada basadas en las fases identificadas, de manera que se cubra el ciclo de vida de la calidad de los datos. Además, se identifican los paquetes y componentes que deben ser modificados en caso de integrar las herramientas independientes a la arquitectura propuesta.

Palabras claves: calidad de metadatos bibliográficos, herramienta integrada, metodologías de calidad de datos, patrones de diseño.

ABSTRACT

In diploma work developed in the period between 2015 and 2017, isolated tools were implemented that provide solutions to some quality problems of the bibliographic metadata cataloged in libraries and documentation centers of the country. However, these alone do not provide an integral view of the entire data quality lifecycle. Therefore, the main objective of the present work is to propose the design of an integrated, extensible and scalable tool that allows the evaluation of the quality of the bibliographic metadata, making use of design patterns and the architectural model Vista Controller Model. To do this, a critical analysis of several data quality methodologies is performed, in order to determine the phases that fit the bibliographic metadata and a critical analysis of the design of the tools implemented so far independently. As a result, we propose the design of an architecture and the visual interface of an integrated tool based on the identified phases, so as to cover the data quality lifecycle. In addition, it identifies the packages and components that must be modified in case of integrating the independent tools to the proposed architecture.

Keywords: Quality of bibliographic metadata, integrated tool, data quality methodologies, design patterns.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La recuperación de la información es uno de los pasos más usuales que realizan los profesionales de la información. Particularmente en bibliotecas y centros de documentación dicho proceso se evidencia a partir de los metadatos que describen los recursos bibliográficos catalogados. Para lograr tales resultados se emplea uno de los sistemas con mayor reputación, la suite para la Automatización de Bibliotecas y Centros de Documentación (ABCD), diseñada por el proyecto BIREME y financiado por el proyecto VLIR (Fernández & Lenzo 2010).

La suite ABCD está dirigida a la gestión integrada de procesos de bibliotecas y operaciones automatizadas. Esta permite el manejo de estructuras bibliográficas y no bibliográficas, además cuenta con cinco módulos principales que ofrecen las funciones de catalogación, préstamos, adquisiciones, registro de publicaciones periódicas (SeCSWeb), y administración de préstamos avanzados (EmpWeb). ABCD trabaja con varios estándares bibliográficos tales como: UNIMARC, CEPAL, AGRIS, MARC 21 (Andreu, 2015, Veloso, 2012).

Por decisión del Ministerio de Educación Superior (MES), en las bibliotecas y centros de documentación cubanos se trabaja con la suite ABCD sin conexión directa a Internet. Esto implica que cada biblioteca cataloga de manera independiente sus registros bibliográficos y para estos fines debe emplear el formato MARC 21, pues constituye uno de los formatos más utilizados en países como Bélgica y España (Medrano et al. 2012).

A pesar de las ventajas que trae consigo su uso, la suite no está exenta de errores que pueden afectar la calidad de los procesos bibliotecarios mencionados anteriormente. La calidad de los metadatos afecta directamente la recuperación de la información (Beall 2006; Borges Zamora 2016).

De manera general, la baja calidad de los datos afecta la reputación de las empresas o entidades poseedoras de los datos, al tomar decisiones basadas en datos erróneos y no fiables. Esta acumulación de errores propicia el aumento de los costos y esfuerzos por mejorar la calidad. Mientras que, datos con elevada calidad conllevan al éxito del negocio y a la satisfacción de los clientes (Andreu Alvarez 2015).

En (Andreu Alvarez 2015) se aplicó una encuesta piloto a 16 especialistas que pertenecen al Centro de Documentación e Información Científico-Técnica (CDICT), que reside en el área académica de la Universidad Central “Marta Abreu” de Las Villas (UCLV). De acuerdo a la opinión de los expertos encuestados se identificaron los principales problemas de calidad presentes en cada módulo (errores ortográficos y tipográficos, registros duplicados y campos nulos o vacíos) y las causas que conducen a los mismos. También se destacó que la entrada manual de los datos y la incorrecta aplicación de las reglas de catalogación angloamericanas son las causas fundamentales que permiten la aparición de errores, siendo el módulo de catalogación el más afectado y a su vez el más utilizado por los especialistas. Este resultado investigativo propició que en tesis posteriores se hicieran aportes en aras de dar soluciones a dichos problemas de calidad. Ejemplos de estas son los trabajos de diploma: (Guevara Torres 2016), que desarrolla la herramienta MARCCompleteness para medir la completitud y detectar duplicados en metadatos con formato MARC 21. En (Lima Ortega 2016) se implementa MARCAccuracy para medir la exactitud en el campo autor del formato MARC 21 utilizando algoritmos de agrupamiento y la distancia EDUK para determinar los autores más parecidos . Mientras que en (Borges Zamora 2016) se elabora la versión 1.0 de la herramienta de perfilado PMMarc para anomalías a nivel de instancias en los campos mínimos del formato MARC 21.

Es importante destacar que estas herramientas brindan soluciones aisladas a varios de los problemas planteados por Andreu Álvarez (2015), aunque por sí solas no logran concluir con una visión íntegra del ciclo de vida de la calidad de los datos, propuesta por Wang (1998) para una mejora continua y una entrega de productos de información de alta calidad (Koronios et al. 2005).

De esta manera se plantea en la presente investigación el siguiente **objetivo general**:

Diseñar una herramienta integrada, extensible y escalable, que permita la evaluación de la calidad de metadatos bibliográficos, haciendo uso de patrones de diseño y del patrón arquitectónico Modelo Vista Controlador (MVC).

Para dar cumplimiento a este objetivo general se proponen los siguientes **objetivos específicos**:

1. Identificar a partir de un análisis crítico de varias metodologías de calidad de datos, las fases que se ajusten a los metadatos bibliográficos.
2. Diseñar una arquitectura que integre las fases identificadas.
3. Diseñar un prototipo de la interfaz visual de una herramienta en Java a partir de la arquitectura propuesta haciendo uso de patrones de diseño y un patrón arquitectónico MVC.

Basándose en dichos objetivos específicos se tomaron en cuenta las siguientes **preguntas de investigación** para el desarrollo de los mismos:

1. ¿Cuáles son las fases de las metodologías de calidad de datos que se ajusten a los metadatos bibliográficos?
2. ¿Qué patrones de diseño utilizar de manera que la herramienta sea extensible y resistente al cambio?
3. ¿Cómo diseñar la interfaz visual de una herramienta en Java siguiendo la arquitectura propuesta?

Justificación de la Investigación:

La investigación posee utilidad práctica debido a que frecuentemente en los centros universitarios del país se catalogan los registros bibliográficos utilizando el formato MARC 21. Hasta el momento sistemas como MARCCompleteness, MARCAccuracy y PMMarc se utilizan para realizar operaciones sobre los metadatos con dicho formato. Estas herramientas dan soluciones aisladas a algunos problemas de calidad, pero ellas por si solas, no brindan una visión íntegra del ciclo de vida de la calidad de los datos, existiendo grandes problemas de calidad. Por tanto, es necesario diseñar una herramienta que integre las ya existentes y modele el ciclo de vida de la calidad de los metadatos bibliográficos. Además, posee una relevancia social porque la investigación se puede extender a otros centros universitarios del país que utilizan el formato MARC 21 para catalogar los metadatos bibliográficos.

Viabilidad:

En el presente trabajo se cuenta con los recursos humanos y materiales necesarios para llevar a cabo la investigación.

Estructura de la tesis:

El presente trabajo de diploma se encuentra estructurado en tres capítulos.

El capítulo I recoge sobre la base de una amplia bibliografía actualizada, los aspectos teóricos que son tratados en la presente investigación. Se analiza cuál es el concepto de calidad de datos, y se hace alusión a autores contemporáneos que lo retoman. Se recoge un análisis crítico a varias metodologías de calidad de datos y se definen a partir de estas, las fases que se ajustan a los metadatos bibliográficos. Se abordan los patrones de diseño empleados y el uso del patrón arquitectónico Modelo Vista Controlador. Además, se da un amplio concepto de que es un sistema integrado, mostrando con este las principales propiedades que posee.

El capítulo II aborda la propuesta de una arquitectura que integrar todo el proceso de calidad de los datos. Se identifican y caracterizan las fases que componen la dicha arquitectura. Se destacan las principales características que presenta la misma.

El capítulo III propone un recorrido por las disímiles herramientas existentes que dan soluciones aisladas a algunos de los problemas planteados por (Andreu Alvarez 2015) y cómo, de una forma u otra, se evidencian estas, en una o varias etapas del ciclo de vida de calidad de la información que propone (Wang 1998). Se realiza una breve caracterización de las herramientas mencionadas y se propone que habría que adaptar en cada una de los sistemas para integrarlas en la arquitectura propuesta.

CAPÍTULO I: ASPECTOS TEÓRICOS

En el presente capítulo se exponen los aspectos teóricos relacionados con el término calidad de datos, así como el formato MARC 21 para describir los metadatos bibliográficos. Además, se realiza un análisis crítico de varias metodologías de calidad de datos y se determinan las fases que se ajustan a los metadatos bibliográficos. Se abordan los aspectos fundamentales de los patrones de diseño seleccionados. Por último, se destacan las principales características que debe cumplir un sistema integrado.

1.1 Definición de calidad de datos

El término de calidad se utiliza frecuentemente como un rasgo imperceptible, algo que puede ser subjetivamente juzgado, pero a menudo no es medido exactamente. *"Términos como buena o mala calidad son intensamente difusos y utilizados sin la intención de ser una ciencia exacta"* (Díaz-de-la-Paz et al. 2015).

Aunque no existe una definición única para el término "calidad de datos" (Bobrowski et al. 1999), una de las más coherentes expuestas en la literatura es la ofrecida por Juran and Gryna (1980), y retomada por autores contemporáneos como (Sadiq et al. 2011; Salomone et al. 2011; Moges et al. 2013; Yeganeh et al. 2014), quienes coinciden en definirla como "datos adecuados para el uso" de los consumidores de datos (García Mendoza 2017; Díaz-de-la-Paz et al. 2015).

En (Wang & Strong 1996) se desarrolla un marco de trabajo donde se organizan las dimensiones de calidad de forma escalonada, tomando como base la opinión de los "consumidores de datos" acerca de qué significaba calidad de datos para ellos, ya que son estos los que deben evaluar si los datos son adecuados o no para su uso.

Precisamente la calidad de los datos depende del contexto de aplicación y del uso que se pretenda de los mismos, por ejemplo los datos pueden ser apropiados para la producción de ventas y sin embargo no ser suficientes para las tareas de contabilidad (Moges et al. 2013).

1.1.1 Análisis crítico de varias metodologías de calidad de datos

En (Batini et al. 2009) se expone que una metodología de calidad de datos se detalla como "*un conjunto de directrices y técnicas que, a partir de una información de*

entrada descrita en un contexto dado, define un proceso racional para evaluar y mejorar la calidad de los datos".

También se plantea que existen varios aspectos que deben de tomarse en consideración para analizar y comparar la calidad de los datos. Entre ellos se destacan:

- las fases y pasos que componen una metodología,
- las estrategias y técnicas que incorpora la metodología para la evaluación y la mejora de los niveles de la calidad de los datos,
- las dimensiones y las métricas que se eligen en la metodología para evaluar los niveles de calidad de los datos,
- tipos de costos asociados al problema de calidad:
    - costos directos (costos de las actividades de evaluación y mejora),
    - costos indirectos (costos asociados con la mala calidad de los datos),
- tipos de datos,
- tipos de sistemas de información.

En cuanto al primer aspecto mencionado, según Batini *et al.* (2009) y Batini and Scannapieco (2016), exponen que una metodología de calidad de datos debe estar sujeta a las fases siguientes:

- Reconstrucción de estado: según Batini et al. (2009) esta fase está encaminada a recopilar información contextual en los procesos organizativos y servicios, recogida de datos y métodos administrativos relacionados, problemas de calidad y costos correspondientes. Puede omitirse esta fase, si se realiza un análisis previo y de estar disponible la información.
- Evaluación/Medición: en esta fase se mide la calidad de la información recogida a través de las dimensiones de calidad y cuenta con los siguientes pasos a seguir: análisis de los datos, análisis de requisitos de calidad de datos, identificar las áreas críticas, modelar el proceso de generación y medición de la calidad (Batini et al. 2009).
- Mejora: en esta fase se seleccionan los pasos, estrategias y técnicas para lograr nuevos objetivos de calidad de datos. Estos pasos se exponen a continuación: evaluación de los costos, asignación de las responsabilidades del proceso y de los datos, identificación de las causas de los errores, selección de estrategias y

técnicas, diseño de soluciones de mejora de datos, control y rediseño de procesos, gestión de la mejora y seguimiento de la mejora (Batini et al. 2009).

El segundo punto expuesto anteriormente, según el autor, expone que las metodologías acogen dos tipos generales de estrategias. Dígase, impulsada por los datos, la cual mejora la calidad de estos cambiando directamente el valor de los mismos, y la impulsada por los procesos, la cual mejora la calidad de los datos al rediseñar los procesos que crean o modifican datos. Estas estrategias aplican una variedad de algoritmos heurísticos y actividades basadas en el conocimiento con el fin de mejorar la calidad de los datos.

En (Batini et al. 2009) se refiere a las dimensiones y métricas de calidad de datos como valores de los datos en comparación con los esquemas. Al analizar una serie de diferencias en la definición de las dimensiones en cuanto al término calidad, propuestas por (Wand & Wang 1996) y (Wang & Strong 1996). Además, es posible definir un conjunto básico de dimensiones de calidad sin dejar de incluir la exactitud, la completitud, la consistencia y el tiempo de vida, que constituyen el foco de atención de la mayoría de los autores (Scannapieco & Catarci 2002).

Al referirse a los costos asociados a los problemas de calidad, se puede concluir que son "*la suma de los costos de las actividades de evaluación de calidad de datos y mejora*" (Batini et al. 2009), conocidos también como costos directos y los indirectos que se revelan como costos asociados a la mala calidad de los datos, la cual puede disminuirse empleando un programa eficaz de calidad de datos, siendo este último más costoso.

En el contorno del término calidad de datos, (Batini et al. 2009) se refiere a tres tipos de datos (ver Figura 1): los estructurados que son agregaciones de los elementos descritos por los atributos elementales definidos dentro de un dominio, los semiestructurados que se expresan como datos que presentan una estructura que posee cierto grado de flexibilidad y los no estructurados definidos como una secuencia genérica de símbolos, codificado típicamente en lenguaje natural.

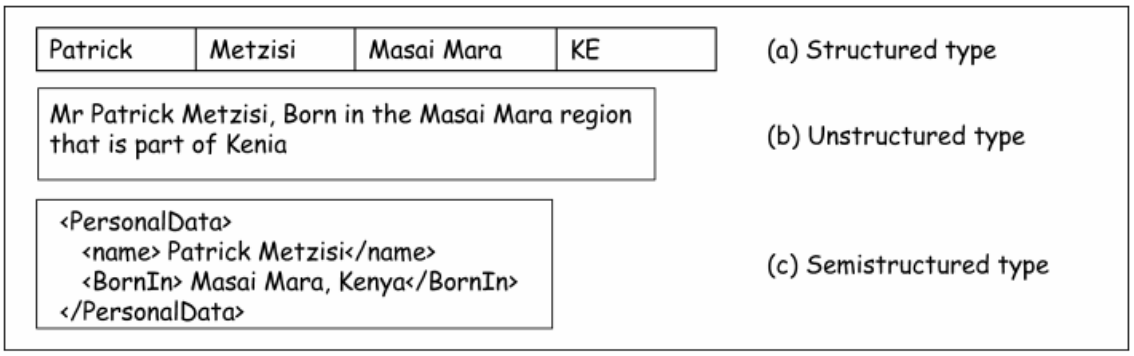

Figura 1. Tipos de datos, un caso del mundo real. Fuente: (Batini et al. 2009).

Los tipos de sistemas de información propuestos por (Batini et al. 2009) difieren sobre el grado de los datos, el proceso y la integración de gestión de apoyo de un sistema técnico, los cuáles se mencionan a continuación:

- sistema de información monolítica,
- almacenamiento de datos (DW, *Data Warehouse*),
- sistema de información distribuida,
- sistema de información cooperativo (CIS, acrónimo de *Cooperative Information System*),
- sistema web de información (WIS, acrónimo de *Web Information System*),
- sistema de información de igual a igual (P2P, acrónimo de *Peer-To-Peer Information System*).

En (Batini et al. 2009) se muestra una comparación entre las metodologías que se muestran en la **¡Error! No se encuentra el origen de la referencia.**, la cual se realiza tomando como base a las dimensiones de calidad de datos, fases, técnicas y estrategias que incluye cada una.

Tabla 1. Metodología para evaluar y mejorar la calidad de los datos. Fuente: (Batini et al. 2009; Batini & Scannapieco 2016).

Acrónimo	Nombre	Referencia
TDQM	*Total Data Quality Management*	(Wang, 1998)
DWQ	*The Datawarehouse Quality Methodology*	(Jeusfeld *et al.*, 1998)
TIQM	*Total Information Quality Management*	(English, 1999)
AIMQ	*A methodology for information quality assessment*	(Lee *et al.*, 2002)
CIHI	*Canadian Institute for Health Information methodology*	(Long y Seko, 2005)
DQA	*Data Quality Assessment*	(Pipino *et al.*, 2002)
IQM	*Information Quality Measurement*	(Eppler y Muenzenmayer, 2002)
ISTAT	*ISTAT methodology*	(Falorsi *et al.*, 2003)
AMEQ	*Activity-based Measuring and Evaluating of product information Quality methodology*	(Su y Jin, 2004, 2006)
COLDQ	*Loshin Methodology (Cost-effect Of Low Data Quality)*	(Loshin, 2001)
DaQuinCIS	*Data Quality in Cooperative Information Systems*	(Scannapieco *et al.*, 2004)
QAFD	*Methodology for the Quality Assessment of Financial Data*	(De Amicis y Batini, 2004)
CDQ	*Comprehensive methodology for Data Quality management*	(Batini y Scannapieco, 2006)

Al analizar los requisitos anteriores y tomando como guía las metodologías expuestas, en (Batini et al. 2009), se destaca que en la fase evaluación los pasos más comunes abordados por las metodologías son el análisis de los datos y la medición de la calidad como se muestra en la Tabla 2. Sin embargo, estos pasos comunes se realizan con distintos enfoques para cada una de las metodologías expuestas.

Tabla 2. Metodologías y pasos de evaluación. Fuente: (Batini et al. 2009; Batini & Scannapieco 2016).

Step/Meth Acronym	Data Analysis	DQ Requirement Analysis	Identification of Critical Areas	Process Modeling	Measurement of Quality	Extensible to Other Dimensions and Metrics
TDQM	+		+	+	+	Fixed
DWQ	+	+	+		+	Open
TIQM	+	+	+	+	+	Fixed
AIMQ	+		+		+	Fixed
CIHI	+		+			Fixed
DQA	+		+		+	Open
IQM	+				+	Open
ISTAT	+				+	Fixed
AMEQ	+		+	+	+	Open
COLDQ	+	+	+	+	+	Fixed
DaQuinCIS	+		+	+	+	Open
QAFD	+	+	+		+	Fixed
CDQ	+	+	+	+	+	Open

Lo mismo sucede con el punto de análisis de requisitos, donde solo las metodologías: DWQ, TIQM, COLDQ, QAFD y la CDQ la consideran, identificando problemas de calidad de datos. Por otra parte, las metodologías: TDQM, TIQM, AMEQ, COLDQ, DaQuinCIS y la CDQ incorporan el modelado de procesos. Destacar que las metodologías que soportan este último aspecto adoptan una estrategia impulsada por procesos en la fase de mejora.

La última columna de la Tabla 2 permite determinar si una metodología dada, brinda la posibilidad de extender o no, a otras dimensiones y métricas.

En (Batini et al. 2009; Batini & Scannapieco 2016) se comparan los pasos de mejora de las diferentes metodologías en cuestión (ver Tabla 3**¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**).

Tabla 3. Metodologías y pasos de mejora (parte 1). Fuentes: (Batini et al. 2009; Batini & Scannapieco 2016)

Step/ Meth. Acronym	Evaluation of costs	Assignment of process responsibilities	Assignment of data responsibilities	Selection of strategies and techniques	Identification of the causes of errors
TDQM	+	+	+	+	+
DWQ	+		+	+	+
TIQM	+	+	+	+	+
DQA					+
ISTAT				+	+
AMEQ					+
COLDQ	+			+	+
DaQuinCIS				+	+
CDQ	+	+	+	+	+

La identificación de las causas de los errores (ver Tabla 3), visto siempre desde diferentes perspectivas, es el paso más tratado en las metodologías. Solo la TDQM, DWQ, TIQM, COLDQ y la CDQ, según Batini *et al.* (Batini et al. 2009), tienen presente la evaluación de los costos, el cual suele ser obligatorio en las metodologías de calidad de datos, ya que se considera un paso crítico para medir la ventaja de las soluciones de mejora y escoger la más eficiente.

En la Tabla 4, se sigue haciendo alusión a los pasos de mejora de las metodologías analizadas. Solo la TDQM, TIQM, ISTAT, COLDQ y la CDQ, presentan el paso de mejora de rediseño de procesos.

Tabla 4. Metodologías y pasos de mejora (parte 2). Fuentes: (Batini et al. 2009; Batini & Scannapieco 2016).

Step/Meth. Acronym	Process control	Process re-design	Improvement management	Improvement monitoring
TDQM		+	+	+
DWQ			+	
TIQM		+		+
DQA				
ISTAT		+		
AMEQ				+
COLDQ	+	+		+
DaQuinCIS				
CDQ	+	+		

Por otra parte, solo dos metodologías de calidad de datos, la COLDQ y la CDQ, presentan el control de proceso, siendo de igual manera para la TDQM y la DWQ que presentan el paso de gestión de la mejora.

1.1.2 Clasificación de las metodologías

En (Batini & Scannapieco 2016) se clasifican las metodologías de calidad atendiendo a varios aspectos:

- Dirigida a la información o dirigida por los procesos.
- De evaluación o de mejora.
- De propósito general o de propósito específico.
- Interorganizativo o Intraorganizativo.

Dirigida a la información o dirigida por los procesos

Según los autores, las estrategias dirigidas a la información se apoyan en el uso exclusivo de fuentes de información para mejorar la calidad de estas. En las estrategias impulsadas por los procesos, el proceso de producción de la información se analiza y posiblemente se modifica para identificar y eliminar las causas fundamentales de los problemas de calidad. En resumen, las metodologías de uso general pueden adoptar estrategias basadas en la información y en el proceso, con diferentes profundidades según la metodología específica.

De evaluación o de mejora

En (Batini & Scannapieco 2016) se hace mención a que las actividades de evaluación y mejora están estrechamente interrelacionadas, ya que sólo cuando las mediciones de la calidad de la información están disponibles es posible concebir las técnicas a aplicar y las prioridades a establecer. Como consecuencia, el límite entre las metodologías de medición y de mejora es a veces impreciso. Se clasifica en el término medición cuando se aborda la cuestión de medir los valores de un conjunto de dimensiones de calidad de información y se cataloga una metodología en el término de evaluación cuando tales mediciones se comparan con valores de referencia, para permitir un diagnóstico de la calidad de la base de información.

De propósito general o de propósito específico

Según Batini and Scannapieco (2016) una metodología se puede clasificar de propósito general cuando cubre un amplio espectro de fases, dimensiones y actividades, mientras que una metodología de propósito específico se centra en una actividad determinada (por ejemplo, medición, identificación de objetos), en un dominio de información específico (por ejemplo, un censo, un registro de direcciones de personas), o en dominios de aplicación específicos (por ejemplo, biología).

Interorganizativo o Intraorganizativo

Se define una metodología intraorganizacional cuando la actividad de medición y mejora se refiere a una organización específica, o a un sector específico de la organización, o incluso a un proceso o base de información específica. De lo contrario, se define como interorganizacional cuando se trata de un grupo de organizaciones (por ejemplo, un grupo de organismos públicos) que cooperan para un objetivo común (por ejemplo, en el caso de las agencias públicas, proporcionando mejores servicios a los ciudadanos y las empresas (Batini & Scannapieco 2016).

1.1.3 Fases que se ajustan a los metadatos bibliográficos

Al analizar minuciosamente la bibliografía referente a las metodologías que plantean (Batini et al. 2009; Batini & Scannapieco 2016) mostradas en la Tabla 1, se puede concluir que las fases de las metodologías que se ajustan a los metadatos bibliográficos

son la fase de análisis, la fase de identificar de áreas críticas y la medición de la calidad, las cuales, se pueden visualizar en la Tabla 2.

Se puede mencionar una primera fase de análisis que cubre todas las metodologías de calidad, debido a que, en esta se examina las bases de información y bases de datos, esquemas y metadatos disponibles en ellos, y realiza entrevistas para llegar a una comprensión completa de la información y las correspondientes reglas de arquitectura y gestión.

Seguidamente se puede hacer alusión a la identificación de áreas críticas como un segundo paso a seguir, debido a que la gran mayoría de las metodologías en cuestión son analizadas, para la selección de las bases y flujos de información más relevantes para ser evaluados cuantitativamente.

Posteriormente como tercera fase se destaca la medición de la calidad, donde se selecciona las dimensiones de calidad afectadas por los problemas de calidad identificados en el paso de análisis de requisitos de la calidad de la información y se define las métricas correspondientes. La medición puede ser objetiva, cuando se basa en métricas cuantitativas; o subjetiva, cuando se basa en evaluaciones cualitativas por parte de administradores y usuarios de información / datos.

La fase de mejora, a pesar de no ser una de las más comunes es imprescindible cuando se desea obtener un buen resultado a la hora de realizar acciones sobre los metadatos bibliográficos. En esta fase se adoptan dos tipos generales de estrategia, las impulsadas por los datos las cuales mejoran la calidad de los datos modificando directamente el valor de los mismos y la estrategia impulsada por procesos, mejoran la calidad al rediseñar los procesos que crean o modifican los datos.

Una fase que no está implícita en todas las metodologías, es la de control, la misma tiene una relevante importancia cuando se desea realizar la auditoria de los datos y determinar así si mejora o empeora la calidad.

Después de analizar minuciosamente las fases comunes e importantes que están presentes en las metodologías de calidad de datos, se toma como guía para la presente investigación a la TDQM; la cual incorpora en su estructura la mayoría de las fases antes mencionadas para medir la calidad.

1.1.4 Metodología TDQM

La metodología TDQM (por sus siglas en inglés de *Total Data Quality Management*) es el resultado de investigaciones académicas y ha sido muy usada como referencia para iniciativas de reingeniería de datos organizacionales (Batini et al. 2009). Esta fue la primera metodología publicada en cuanto a calidad de datos se refiere (Wang 1998) y utiliza una versión del ciclo planteado por W. E. Deming (1986), el cual se identifica por cuatro pasos principales: planificar, hacer, verificar y actuar.

En años posteriores, (Wang 1998) realiza una semejanza entre la TQM (*Total Quality Management*) y la TDQM; donde expone que la fabricación de un producto puede ser vista como un sistema de procesamiento (Díaz-de-la-Paz et al. 2015). Análogamente, la fabricación de información puede ser vista como un sistema de procesamiento que opera sobre los datos en bruto para producir productos de información (Koronios et al. 2005), como se muestra en la Tabla 5.

Tabla 5. Fabricación de productos contra fabricación de información. Fuente: (Wang 1998).

	Fabricación de Productos	**Fabricación de Información**
Entrada	Materiales sin pulir	Datos en bruto
Proceso	Línea de ensamblaje	Sistema de Información
Salida	Productos físicos	Productos de Información

Según, Wang (1998), el sistema de fabricación de información se propuso para: definir, medir, analizar y mejorar los productos de información (ver Figura 2).

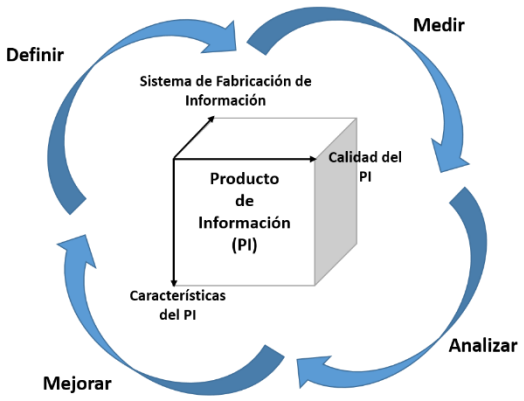

Figura 2. Ciclo de vida de la metodología TDQM. Fuente: (Wang 1998)

La mayoría de las metodologías presentadas por (Batini et al. 2009; Batini & Scannapieco 2016) y dentro de la que se encuentra la TDQM, están dirigidas a datos estructurados, las fases de estas, pueden ajustarse a datos semiestructurados, como es el caso de los metadatos bibliográficos, solo que varía la manera de darle solución a las problemáticas que se presentan en cada fase.

1.2 Metadatos bibliográficos

Según Caplan (1995), el término metadatos fue expuesto por Jack Myers en la década de los 60 para detallar conjuntos de datos. Existen diferentes acepciones sobre metadatos, siendo la más sencilla y actualmente la más extendida la de "*datos sobre los datos*", debido a que proporcionan la información mínima necesaria para identificar un recurso. Autores como (Xu 1998; Dempsey & Heery 1998; Tennant 1998; Taylor 2004; Caplan 1995), amplían este concepto al afirmar que los metadatos también deben incluir información sobre el contexto, contenido y control. Así lo reafirman otras definiciones como la que defiende (Xu 1998), donde sostiene este término como un conjunto de elementos que pueden ser usados para describir y representar objetos de información independientemente del medio o soporte. (Dempsey & Heery 1998) los definen como datos que describen los atributos de un recurso, mientras que (Tennant 1998), la precisa como "información estructurada sobre información".

Por otra parte, en (Caplan 1995) se considera a las fichas de los catálogos tradicionales como metadatos ya que poseen datos bibliográficos (dígase autor, título, editorial, etc.) que se refieren a otros documentos. Además, en (Taylor 2004), se introducen dos nuevos conceptos en cuanto a metadatos se refiere, el de contenido y el de codificación. Ampara que cuando existe contenido, se le denomina registro bibliográfico, mientras que cuando hay codificación se identifica como una estructura.

1.2.1 Formato MARC 21

Según Medrano, Figuerola y Alonso (2012), el estándar bibliográfico MARC 21, constituye uno de los formatos más empleados en países como Bélgica y España. Es un modelo de metadatos utilizado para la representación e intercambio de datos bibliográficos, el cual está compuesto de tres componentes principales: cabecera, directorios y campos variables (ver Figura 3).

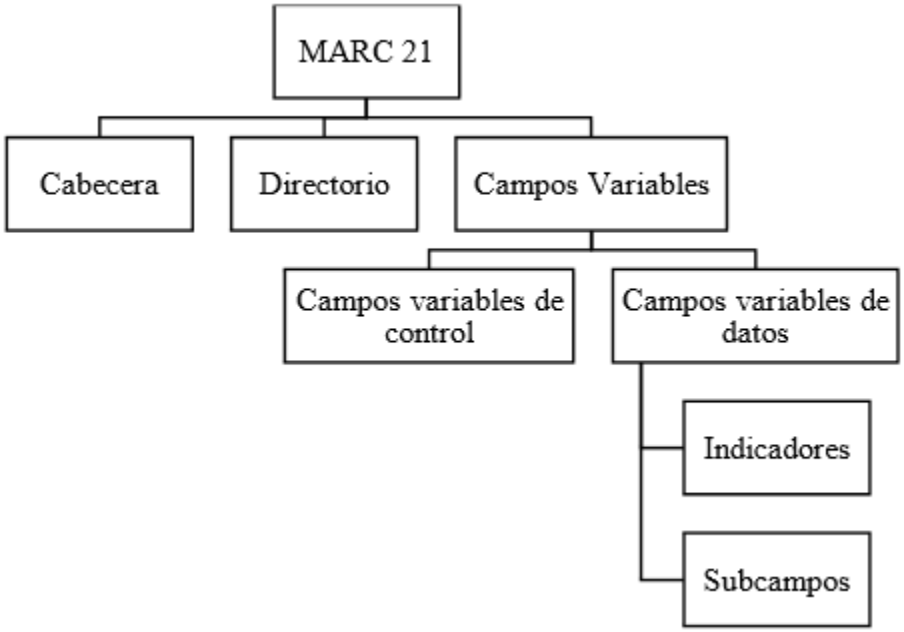

Figura 3. Estructura de un registro MARC 21. Fuente: (García-Mendoza et al. 2016)

Según García Mendoza (2017), la estructura correspondiente al formato MARC 21 es la siguiente:

- Cabecera: es el primer campo del registro, que comprende las primeras 24 posiciones (comenzando por 00) y este provee información para el procesamiento del mismo.
- Directorio: es el segundo campo que aparece inmediatamente a continuación de la cabecera en la posición 24. Consiste en una serie de entradas de longitud fija (12 caracteres) que indican:
    - tiqueta: ocupa tres caracteres numéricos que identifican un campo asociado (ver **¡Error! No se encuentra el origen de la referencia.**).
    - Longitud del campo: ocupa cuatro caracteres numéricos que indican la longitud del campo, incluyendo los indicadores, los códigos de los subcampos, los datos y el elemento que indica el final del campo. El número se justifica a la derecha y las posiciones no utilizadas toman valor cero.
    - Posición del carácter final: ocupa los últimos cinco caracteres numéricos que indican la posición del carácter inicial del campo en relación al inicio de los datos de la cabecera del registro.

- Campo variable: son los encargados de organizar los datos de un registro en el estándar MARC 21. Cada campo se identifica por una etiqueta numérica

de tres caracteres que se almacena en la entrada del directorio para el campo correspondiente (ver Figura 3). Los campos repetibles son los que pueden tener varias instancias, por otro lado, los no repetibles son los que pueden tener solamente una instancia. Estos campos variables pueden ser:

- De control: campos que su etiqueta tiene la forma 00X. Estos campos de control son diferentes a los campos variables de datos, en cuanto a estructura se refiere, debido a que no contienen indicadores ni subcampos. Contienen un elemento individual de datos o una serie de elementos de longitud fija identificados por su posición relativa.
- De datos: al principio de cada campo para los dos indicadores, contiene dos posiciones, una para cada uno. Las restantes posiciones del campo contienen varios subcampos. Estos últimos tienen al inicio un código de dos caracteres (un delimitador y un carácter alfanumérico en minúscula, dígase letra o número) y seguido los datos. Estos subcampos pueden ser repetibles y no repetibles con respecto a un campo determinado.

Tabla 6. Función de cada grupo de etiquetas.[1]

Grupo de etiquetas	Función
0xx	Números de control bibliográfico e información codificada.
1xx	Entradas principales.
2xx	Información sobre títulos, edición. sello editorial, etc.
3xx	Descripción física, etc.
4xx	Menciones de series.
5xx	Notas.
6xx	Entradas de acceso por tema.
7xx	Entradas agregadas que no son de tema, ni de series ni campos de enlace.
8xx	Entradas agregadas de series y existencias
9xx	Campos para uso local.

Para el estándar MARC 21 para datos bibliográficos existen ocho clases de materiales (CM): libro (L), archivo de computadora (A), partitura (P), mapa (M), publicación seriada (Ps), material mixto (Mm), material visual (Mv) y grabación sonora (Gs) (Styles et al. 2008).

Estas clases de materiales se determinan a través de las posiciones seis y siete de la cabecera de un registro. La posición seis indica el tipo de material y los valores permitidos se pueden apreciar en la columna “Código de tipo” (ver Tabla 7).

[1] https://www.oclc.org/bibformats/es/introduction.html

Tabla 7. Clases de material del formato MARC 21 para datos bibliográficos.[1]

CM	Código de tipo		Nivel bibliográfico
L	a	Material lingüístico	a, c, d, m
	t	Material lingüístico manuscrito	a, c, d, m
Ps	a	Material lingüístico	b, i, s
Mv	g	Medio proyectado	a, b, c, d, i, m, s
	k	Gráfico bidimensional no proyectable	a, b, c, d, i, m, s
	r	Artefactos y objetos naturales tridimensionales	a, b, c, d, i, m, s
	o	Kits	a, b, c, d, i, m, s
Mm	p	Material mixto	c, d
M	e	Material cartográfico	a, b, c, d, i, m, s
	f	Mapa manuscrito	a, c, d, m
P	c	Música impresa	a, b, c, d, i, m, s
	d	Música manuscrita	a, c, d, m
Gs	i	Grabación sonora no musical	a, b, c, d, i, m, s
	j	Grabación sonora musical	a, b, c, d, i, m, s
A	m	Archivo de computadora	a, b, c, d, i, m, s

Por otro lado, la posición siete representa el nivel bibliográfico (ver Tabla 8).

Tabla 8. Niveles bibliográficos que pueden aparecer en la posición siete de la cabecera..[1]

Código	Nivel bibliográfico
a	Componente, monografía.
b	Componente, publicación seriada.
c	Colección.
d	Subunidad.
i	Recurso integrado.
m	Monografía.
s	Publicación seriada.

1.3 Patrones de diseño

Los patrones de diseño son descripciones de clases y objetos relacionados que están adaptados para resolver un problema de diseño general en un contexto determinado (Gamma et al. 1995), por otra parte (Grand 2002), define los patrones de diseño, como una solución a problemas recurrentes que surgen en el desarrollo de un software, esta solución es reusable para otros problemas similares. El uso de patrones de diseño permite tener una solución implementada cuando un desarrollador de software se encuentre con un problema determinado.

1.3.1 Patrón *Adapter*

La intención del patrón *Adapter* es trasformar la interfaz de una clase en otra interfaz esperada por los clientes sin cambiar su funcionalidad básica. Permite a las clases trabajar juntas, lo que de otra manera no podría hacerse debido a sus interfaces incompatibles. Por ejemplo, permite la interoperabilidad entre un paquete de geometría que requiere ángulos para ser especificados en radianes y un cliente que espera pasar ángulos en grados (Fall 2005).

La Figura 4 muestra la estructura del patrón, donde una clase adaptadora usa múltiples herencias para adaptar una interfaz a otra.

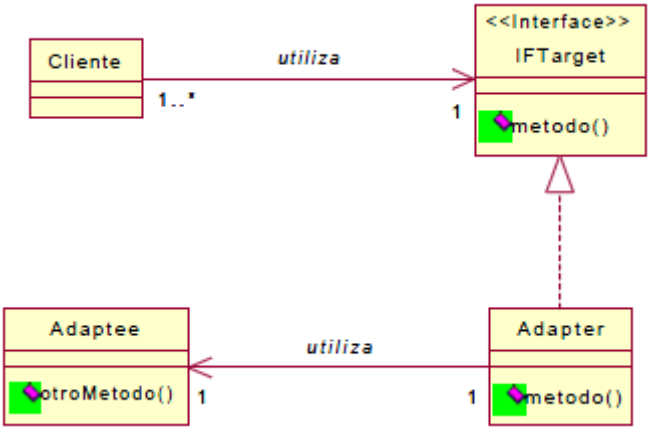

Figura 4. Uso del patrón Adapter. Fuente: (Martínez Juan & Cueva Lovelle 1999).

En este caso especial, al utilizar el patrón *Adapter*, la clase "Cliente" llama a un método de otra clase a través de una interfaz. En la interfaz "*IFTarget*" se declara el método que la clase "Cliente" necesita. Por otra parte, en "*Adapter*" se implementa la interfaz "*IFTarget*", por lo que en este se debe desarrollar el método que el cliente necesita para llamar a un método de la clase "*Adaptee*". Mientras que, en la clase "*Adaptee*", no se implementa el método de la interfaz "*IFTarget*", pero tiene un método que necesita la clase "Cliente".

1.3.2 Patrón *Bridge*

El patrón estructural *Bridge* o Puente surge con el objetivo de desacoplar una abstracción de su implementación de forma que cada una de ellas pueda variar independientemente, es decir que permita el cambio en tiempo de ejecución (Gamma et al. 1995).

De manera general se suele emplear la herencia para realizar la implementación de abstracciones, lo que trae consigo que se asocien de forma permanente la abstracción y la implementación.

Al tomar como ejemplo (ver Figura 5) un entorno de ventanas que pueden funcionar en dos sistemas diferentes (por ejemplo, MAC y *Windows*), la abstracción *Windows* permite desplegar aplicaciones que funciones en los dos sistemas, pero la implementación de la misma deberá ser distinta en cada sistema.

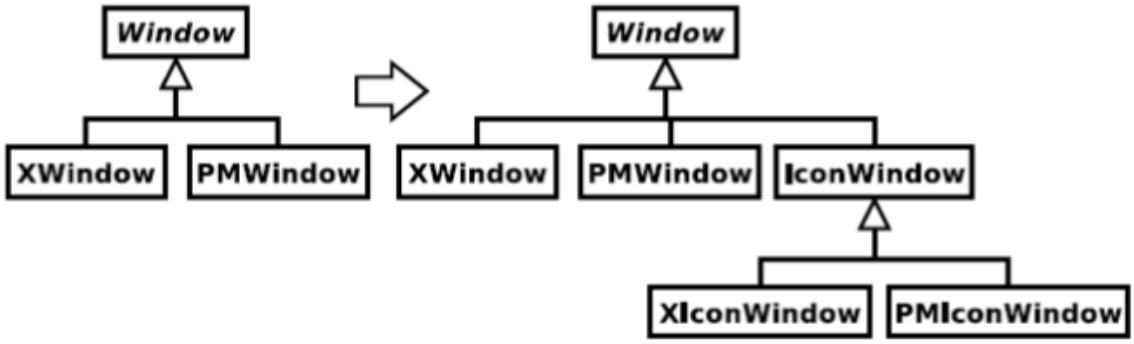

Figura 5. Ejemplo para aplicar el uso del patrón Bridge. Fuente:(Gamma et al. 1995).

Como una solución viable al inconveniente anterior, seria implementar la clase abstracta "*Window*" con clases hijas que implementen la abstracción en cada entorno, la cual genera los siguientes problemas:

- extender la clase abstracta implica modificar cada una de las subclases,
- existencia de problemas de dependencia en la plataforma.

Otra solución factible sería separar en dos jerarquías la clase abstracta "*Window"* y sus implementaciones independientes de la plataforma.

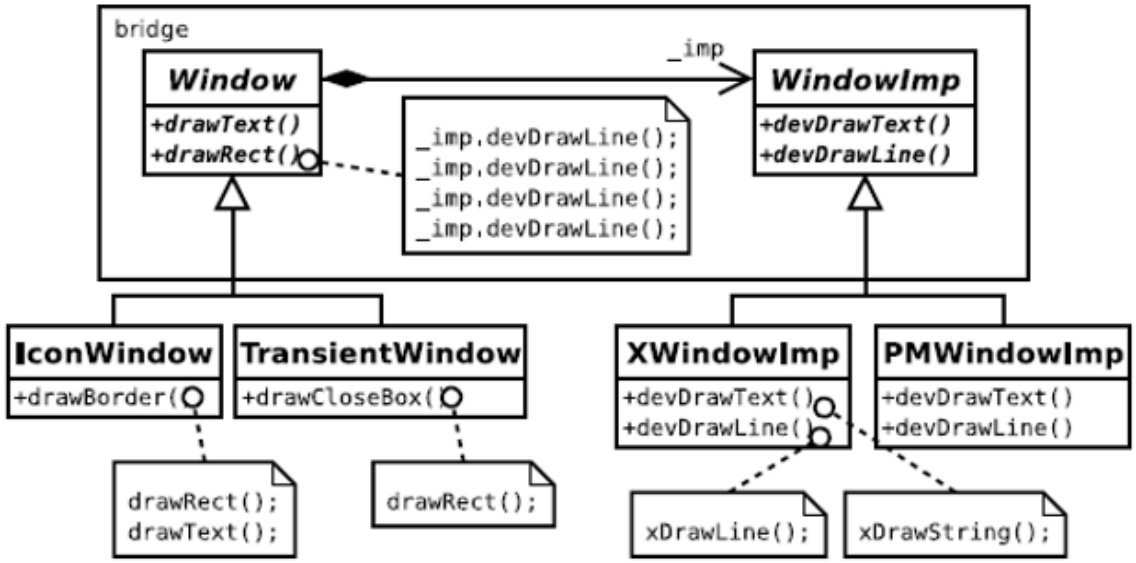

Figura 6. Uso del patrón Bridge. Fuente:(Gamma et al. 1995).

De este modo todas las operaciones utilizadas en las clases hijas de "*Window"* se definen como métodos abstractos de la interfaz "*WindowImp"*.

La estructura final del patrón es el que se visualiza en la Figura 7, donde las clases participantes son "*Abstraction"*, quien define la interfaz de la abstracción y mantiene la referencia al objeto implementador. La clase "*RefinedAbstraction"* extiende la interfaz definida por la abstracción. "*Implementor"* precisa la interfaz para los implementadores concretos ("*ConcreteImplementorX"*), esta no tiene por qué corresponderse con la abstracción.

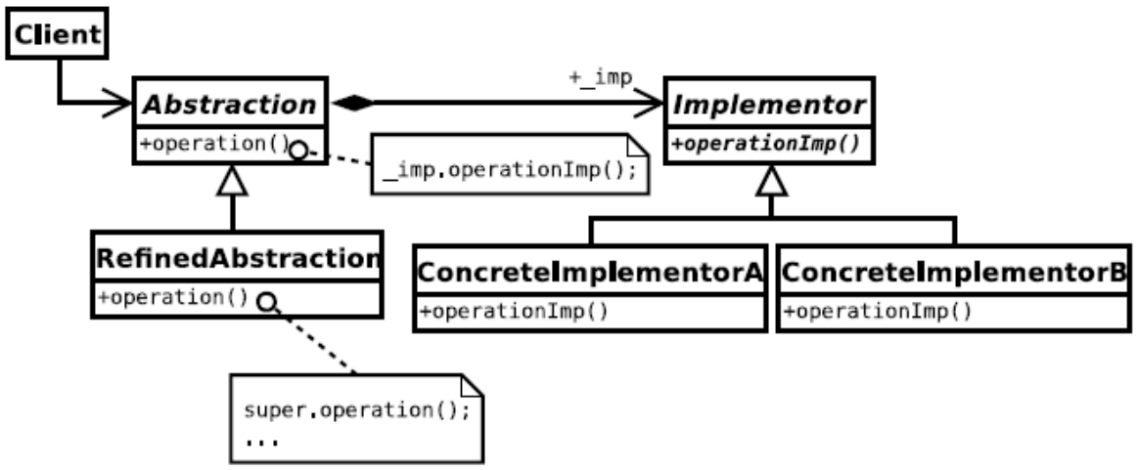

Figura 7. Estructura del patrón Bridge. Fuente: (Gamma et al. 1995)

1.3.3 Patrón *Strategy*

El patrón *Strategy* permite elegir dinámicamente un algoritmo en tiempo de ejecución. Estos algoritmos o clases desarrollan la misma interfaz y mediante el polimorfismo pueden ser tratados como un mismo tipo (Grand 2002). Según Guevara Torres (2016), el patrón facilita la implementación de distintos comportamientos específicos en clases descendientes a través de una misma clase.

Strategy se puede aplicar a subclases o interfaces en dependencia de la preferencia del programador o la conveniencia de donde se implemente, esto también facilita la adición de nuevas estrategias sin tener que modificar el código existente (Larman 1999).

Los múltiples componentes que encierran al referido patrón se despliegan a continuación:

- Interfaz *Strategy*: en la misma se incorpora el nombre de los métodos (este puede ser uno o varios) que conforman la estrategia.
- Clases *Strategy* específicas: lo satisface en todas las clases que implementan la interfaz anterior.
- Contexto: elemento donde se desenvuelve la estrategia[2].

En la Figura 8 se ilustra un diagrama de clases diseñado por (Guevara Torres 2016) donde se utiliza el patrón Strategy.

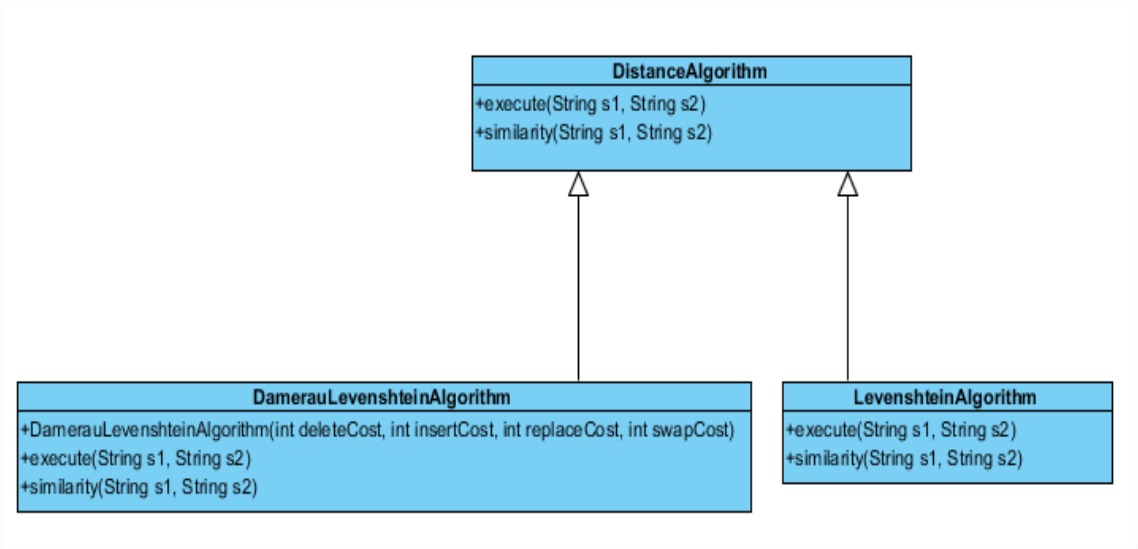

Figura 8. Uso del patrón Strategy. Fuente:(Guevara Torres 2016).

[2] http://www.seas.es/blog/informatica/patrones-de-diseno-en-java-patron-strategy/.

1.4 Integración de software

La integración de componentes en un proyecto es un problema complejo, sobre todo en sistemas que involucran código desarrollado por distintas personas, por eso es necesario contar con un entorno que garantice la adecuada integración de las partes y permita mostrar los resultados de la integración de una manera fácil y sencilla.

1.4.1 Patrón arquitectónico MVC

El patrón arquitectónico MVC (Modelo-Vista-Controlador), surge con la intensión de dividir el modelado del dominio, la presentación y las secciones basadas en las entradas del usuario en tres fases (Burbeck 1992).

Estas etapas se visualizan en la Figura 9, donde el modelo (*Model*) trata el comportamiento, los datos del dominio de la aplicación, y responde a los requerimientos de información acerca de su estado y a las instrucciones para cambiar de estado.

La vista (*View*) maneja el despliegue de la información, o sea, gestiona la salida gráfica y/o textual de la pantalla, mientras que el controlador (*Controller*) se emplea para enviar mensajes al modelo y proporciona la comunicación entre el modelo con sus vistas asociadas, y los dispositivos interactivos de interfaz de usuario como son el *mouse* y el teclado (Krasner & Pope 1988).

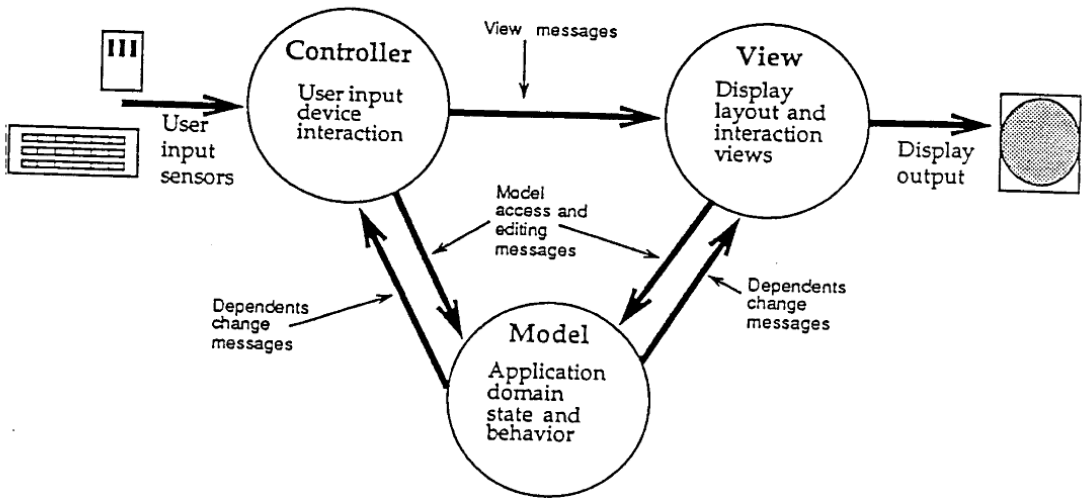

Figura 9. Patrón arquitectónico MVC. Fuente: (Krasner & Pope 1988).

Tanto la vista como el controlador dependen en gran medida del modelo, no sucediendo así en orden inverso. Esta separación posibilita que el modelado sea construido independientemente de la presentación visual.

1.5 Propiedades del sistema integrado

En los últimos años la tendencia actual del desarrollo de software exige cada vez más que se abarquen una amplia gama de aspectos, no solo centrándose en un solo objetivo sino incorporando otros. Características como la escalabilidad, la resistencia al cambio y la extensibilidad, son funcionalidades precedentes que todo sistema debe poseer para que su rendimiento sea óptimo. En el presente trabajo se aborda sobre algunos de estos aspectos funcionales.

1.5.1 Escalable

La escalabilidad es un concepto que existe desde siempre en el mundo informático, aunque parezca una opción clara, es un aspecto complejo e importante. La misma está íntimamente ligada al diseño del sistema e influye en el rendimiento de forma significativa. Si este esquema está bien trazado, la escalabilidad no constituye un problema, pues esta supone un factor crítico en el crecimiento del mismo.
Por tanto un sistema escalable se define como la capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa la carga de sus operaciones, sin perder calidad ni prestaciones (Méndez Menéndez 2014).
Usualmente se evidencian dos tipos de escalabilidad:

- Vertical
- Horizontal

La escalabilidad vertical o lo que es lo mismo escalar hacia arriba, significa añadir más recursos a un solo nodo en particular dentro de un sistema, es decir, añadir un disco duro más rápido a una computadora

Por otro lado, la escalabilidad horizontal representa agregar más nodos a un sistema determinado, es decir, añadirle una computadora a un sitio web que contiene una réplica exacta de otra (*mirror*).

1.5.2 Resistente al cambio

Existen innumerables acepciones en cuanto al termino resistencia se refiere, una de estas es la oposición a la acción de una fuerza y la capacidad de resistir.[3] Por lo que se define que un sistema resistente al cambio es todo aquella herramienta que resista y mantenga las mimas prestaciones cuando un usuario avanzado le realice alguna modificación parcial o total a su implementación.

1.5.3 Extensible

En (Ledesma 2004), se define que un sistema es extensible, cuando permite que un usuario avanzado pueda expandir las capacidades del programa. Este es un aspecto primordial que debe presentar todo sistema es ser extensible, permitiendo que a lo largo del tiempo se incorporen nuevas funcionalidades (dígase, nuevos estándares bibliográficos como *DublinCore*, nuevas extensiones de formato de bases de datos, nuevas métricas, nuevas dimensiones, etc.). Esto permite a gran medida que el sistema sea usado por los especialistas, brindándoles a estas nuevas tareas de manera automatizada.

1.6 Conclusiones parciales

En el presente capítulo se realizó una revisión de la bibliografía sobre aspectos teóricos importantes como la calidad de datos y el formato MARC 21 para describir los metadatos bibliográficos. Se recurrió a los aspectos planteados en (Batini et al. 2009; Batini & Scannapieco 2016) para formalizar, caracterizar y analizar críticamente las metodologías de calidad de datos. A partir de lo cual, se identificaron las fases más comunes a todas las metodologías y se determinaron cuáles de estas se ajustan a los metadatos bibliográficos. Además, se expusieron las principales ventajas y características que presentan los patrones de diseño *Adapter*, *Strategy* y *Bridge* para la implementación de un sistema integrado. Finalmente, se abordan aspectos deseables en este tipo de sistema tales como la escalabilidad, la resistencia al cambio y el carácter extensible.

[3] http://todosobrecambiosorganizacionales.blogspot.com/2010/04/resistencia-al-cambio.html

CAPÍTULO II: ARQUITECTURA INTEGRADA DE UNA SOLUCIÓN DE CALIDAD DE DATOS PARA METADATOS BIBLIOGRÁFICOS

En el presente capítulo se aborda en función de los metadatos bibliográficos, la arquitectura que se propone para integrar todo el proceso de calidad de los datos. Se identifican y caracterizan las fases que componen la arquitectura propuesta, así como las principales características que presenta la misma.

2.1 Diseño de una arquitectura integrada

Existen herramientas que brindan soluciones aisladas a algunos de los problemas de calidad planteados en (Andreu Alvarez 2015), pero no se puede apreciar un sistema que muestre por sí misma, todo el proceso de vida de la calidad de los metadatos bibliográficos. Para dar solución a esta interrogante se propone una arquitectura que integre todo el proceso antes mencionado, dando paso a un sistema que incorpore las herramientas expuestas (ver Figura 10).

Para diseñar la arquitectura propuesta se toma como referencia el ciclo de vida de calidad de la información propuesto por (Wang 1998), retroalimentado con una fase de control.

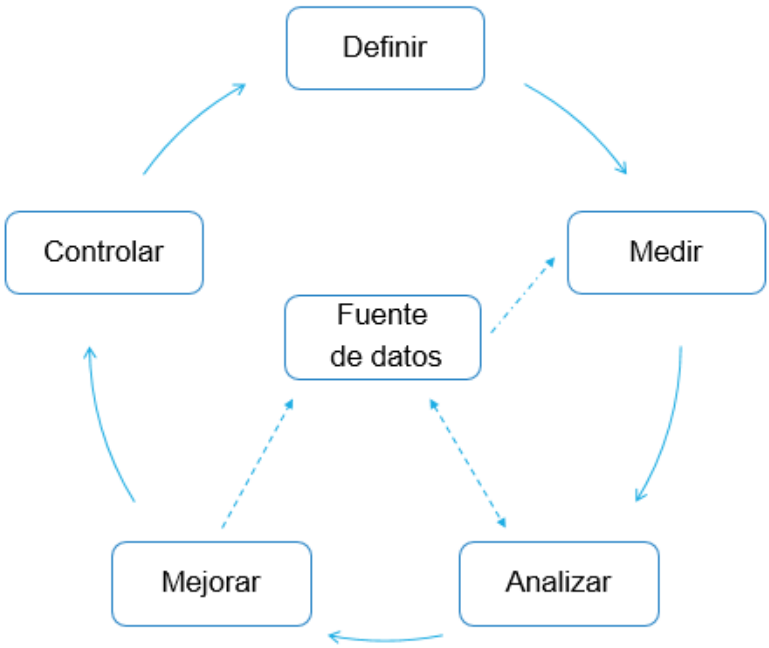

Figura 10. Arquitectura propuesta que integra todo el proceso de calidad de los metadatos bibliográficos. Fuente: (Elaborado por el autor).

Donde:

 Representa la lectura de una fase determinada sobre de las fuentes de datos

 Representa la modificación (o escritura) en las fuentes de los datos de una fase determinada

 Representa la combinación de los anteriores, es decir la lectura y escritura de una fase sobre las fuentes de datos.

2.1.1 Fases que componen la arquitectura

Las fases que se proponen en la arquitectura para integrar todo el proceso de calidad de la información son: Definir, Medir, Analizar, Mejorar y Controlar.

En las investigaciones realizadas “Definir” es una fase preconcebida, es decir, se toma como referencia las siete dimensiones generales o independientes del dominio que se plantean en (Bruce & Hillmann 2004), con el objetivo de evaluar la calidad de los metadatos. Estas dimensiones son: completitud, exactitud, conformidad a las expectativas, consistencia lógica, accesibilidad, tiempo de vida y procedencia. En el presente trabajo solo se abordarán las dos primeras dimensiones (completitud y exactitud), pues son las más tratadas por los expertos y las definidas en los trabajos precedentes.

En la fase “Medir” se realizan los cálculos correspondientes a las métricas, específicamente en todo el registro en la dimensión completitud, mientras que en la dimensión exactitud, se calcula para el campo autor.

En la fase “Analizar” se ejecuta el perfilado de los datos, detectando las anomalías descriptivas de los mismos, determinando su clasificación, se detectan datos duplicados y se estandarizan los valores. Dicho módulo (fase Análisis), permite un gran número de funcionalidades, dentro de las que se puede hacer referencia:

- Detectar cabeceras incorrectas.
- Revelar idiomas inciertos.
- Determinar años de publicación erróneos.
- Estipular la cantidad de campos no nulos y valores mínimos por tipo de material.

- Realizar reportes para detallar los diferentes resultados obtenidos, después de aplicar una herramienta de perfilado.

En la fase “Mejorar” se realiza la limpieza de los datos, detectando duplicados e integrando los datos de los mismos. Esta etapa es primordial cuando se desean tener valores con una elevada calidad.

Finalmente, en la fase de “Control” se ejecutan las tareas de auditoría, se realiza el análisis y reporte de los datos, se almacenan dichos resultados en un repositorio con los metadatos de control. Este módulo está presente en las fases antes mencionadas, pues en él, se van a registrar todas las acciones que se realicen en cada etapa del ciclo de vida de la calidad de los datos. Es esencial, para tener un control estricto de los resultados que se ejecutan sobre las colecciones, para así determinar cuál es el más efectivo, es decir, si empeora o mejora la calidad de los metadatos.

2.2 Principales características

La arquitectura que se propone como solución de la problemática de la presente investigación, está estructurada en paquetes (ver Figura 11). La misma cuenta con un paquete “*dimension”*, donde se agrupan las siete dimensiones que se plantean en (Bruce & Hillman, 2004) para medir la calidad, (en la Figura 11 solo se hace alusión a la exactitud y a la completitud, por ser las más tratadas por los expertos). A su vez, cada una de estas dimensiones, contienen un subpaquete “*metrics”*, que es donde se van a ubicar las diversas métricas de calidad.

Además, contiene un paquete para la entrada y la salida de los datos. Además, se puede apreciar el paquete controlador de las vistas de la arquitectura (se refiere a “*view”*) para una mejor visualización de los resultados obtenidos. Incorpora, además una estructura de paquetes que permite incorporar los estándares bibliográficos existentes, en la presente investigación, solo se refiere al formato MARC 21.

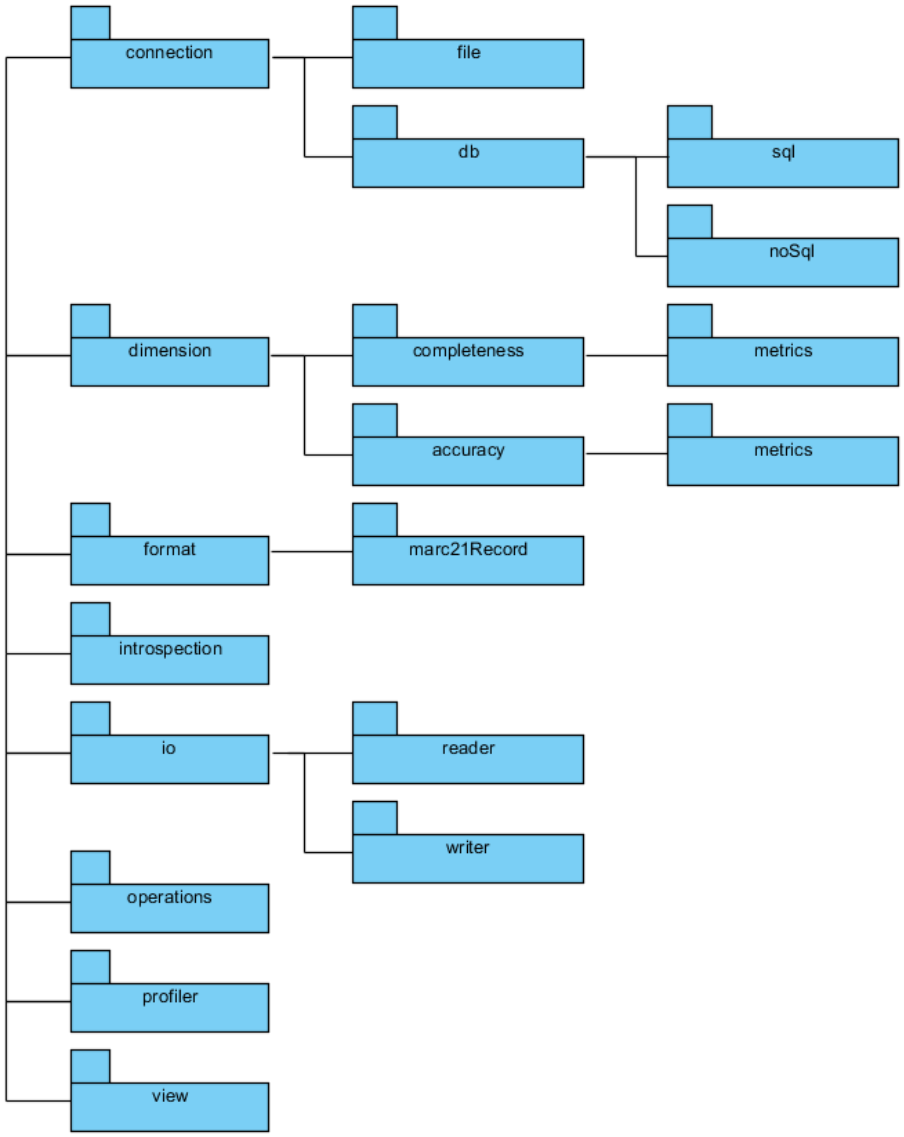

Figura 11. Diagrama de paquetes, de la arquitectura propuesta. Fuente: (Elaborado por el autor)

Como propuesta para incorporar y añadirle nuevas funcionalidades a la arquitectura en posibles versiones futuras, se emplea el uso de la introspección en la conexión una colección determinada, para ello se muestra en Figura 12 el diagrama de clases respectivo.

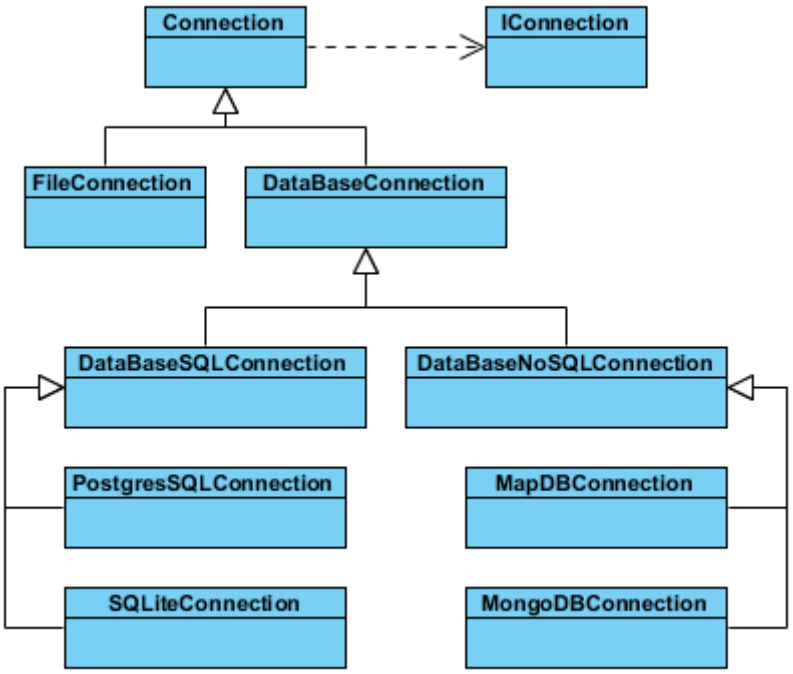

Figura 12. Diagrama de clases, utilizando la introspección en el paquete connection. Fuente: (Elaborado por el autor)

2.3 ¿Qué es la introspección?

La introspección es la facilidad que presentan algunos lenguajes orientados a objeto para determinar un tipo de objeto en tiempo de ejecución y una característica común en todo lenguaje que permita que clases de objetos sean manipuladas como objetos de clase por el programador.

Facilidades que permite el uso de la introspección:

- Acceso en tiempo de ejecución a las clases, así como a los métodos y atributos definidos en cada una de ellas.
- Hace extensible al sistema que se desarrolla, dando la posibilidad de agregarle funcionalidades sin tener que modificar el código anterior.

En el paquete antes mencionado (“*introspection”*) se propone, la clase “*Introspection”*, la cual permite el acceso a las clases de un paquete. Esta búsqueda se realiza tanto en un paquete determinado como en subpaquetes de este. En la estructura que se propone solo se itera en un determinado paquete y no se realiza la búsqueda en subpaquetes por la organización de las clases que se propone.

Otra característica notable de la clase referida, es que permite obtener todos los atributos y métodos de una clase determinada, así como establecer los parámetros que necesitan para ejecutarse y decretar el tipo de dato que devuelven los métodos o el tipo que es un atributo específico.

Una destacada funcionalidad, es que permite instanciar en tiempo de ejecución una clase concreta, comprobar cuál es la clase de la que extiende, si es o no, una clase abstracta o una interfaz.

Para facilitar el acceso a las colecciones, la arquitectura que se propone, cuenta con un paquete "*connection"*, la cual contiene 5 clases fundamentales que permiten el acceso a los datos. "*IConnection"*, es una interfaz, de la cual tienen que implementar cada una de las clases específicas ya que contiene los métodos *isConected(), getNameConection(), connect()* y *close()* que deben tener todas las conexiones. "*Connection"* es una clase abstracta que contiene los atributos que son comunes en todas las conexiones: nombre, tipo de conexión y si esta está establecida o no. En "*ComponentsType"* se definen los parámetros de forma general (servidor, puerto, usuario, contraseña, entre otros), que permiten establecer la conexión con una colección determinada, facilitando con esto la extensibilidad de la herramienta. En "*ConstantConnectionType"* se define un tipo de constante para una colección que se desee agregar al sistema, permitiendo con esto su extensibilidad a otras opciones.

El acceso a una colección, se dividen en dos grupos, mediante un fichero MARC, el cual se define en "*FileConnection"* y a través de la clase abstracta "*DataBaseConnection"*, que extienden de "*Connection"*. De "*DataBaseConnection"* extienden las clases abstractas "*DataBaseSQLConnection"* y "*DataBaseNoSQLConnection"*, las cuales hacen referencia a la lista de base de datos SQL y no SQL de las cuales se pueden tomar los datos.

Procedimiento para incorporar una base de datos a la arquitectura:

Al realizar estos pasos, facilita que el sistema cargue automáticamente por introspección los parámetros necesarios para establecer la misma:

- **Paso 1:** Determinar si la nueva base de datos a incorporar es SQL. En caso afirmativo ubicarla en la ruta "*connection.db.sql", y* en caso contrario "*connection.db.nosql".*

- **Paso 2:** El nombre la clase, debe comenzar con el seudónimo que se desea visualizar en la aplicación, seguido de la palabra "*Connection*". Por ejemplo, si se desea incorporar PostgreSQL a la lista de colecciones, el nombre de la clase debe ser "*PostgreSQLConnection*".
- **Paso 3:** Dicha clase tiene que extender de *DataBaseSQLConnection si es SQL y en caso contrario extiende de DataBaseNoSQLConnection.*
- **Paso 4:** Se debe agregar una nueva constante de la clase creada, para ello, ir al paquete *connection*, específicamente a la clase *ConstantConnectionType* e incorporarla.
- **Paso 5:** Al extender la nueva clase de *IConnection*, esto obliga a que se deban implementar todos sus métodos. Se debe crear un constructor vacío, permitiendo con esto, que la nueva clase sea utilizada por el visual en tiempo de ejecución.
- **Paso 6:** Añadir en la nueva clase agregada, la lista de parámetros necesarios para establecer la conexión, en el método *getParameters()* de la siguiente forma:
    - Se crea una lista de parámetros como se muestra a continuación:
      ArrayList<Parameters> l = new ArrayList();
    - Si se desea agregar a la conexión un servidor, solamente se debe incorporar la línea siguiente:
      l.add(new Parameters(ComponentsType.HOST));
      Lo mismo sucede con cualquiera de los parámetros definidos en la clase *ComponentsType*, que se ubica en el paquete *connection.* En caso de un nuevo parámetro que no exista en dicha clase, agregar este a la clase anterior e incorporar en el visual los componentes adecuados.
    - Para culminar, se retorna la lista de parámetros.

2.4 Conclusiones parciales

En el presente capítulo se definió la propuesta de la arquitectura que integra todo el proceso de calidad de los datos, tomando como base las fase comunes e importantes que presentan las diversas metodologías de calidad de los datos. Se describió las particularidades de cada fase, destacando las ventajas de su uso. Se presentó las principales características de la arquitectura propuesta, resaltando el diagrama de paquetes que se propone, para integrar las herramientas existentes. Se propuso el uso de

la introspección al establecer una conexión, demostrando con esto, la extensibilidad de la arquitectura.

CAPÍTULO III: DISEÑO DE LA HERRAMIENTA INTEGRADA

En el presente capítulo se propone un recorrido por las diversas herramientas existentes que dan soluciones aisladas a algunos de los problemas planteados por (Andreu Alvarez 2015) y que de una forma u otra se evidencia en una o varias etapas del ciclo de vida de calidad de la información que propone (Wang 1998). Se realiza una breve caracterización de las herramientas mencionadas y se identifican las partes que se deben adaptar en cada uno de los sistemas para ser integradas a la arquitectura y al diagrama de paquetes que se propone.

3.1 Caracterización de las herramientas existentes

Para incorporar las herramientas precedentes a la arquitectura propuesta, se toma como paso inicial una breve caracterización de la estructura de los diseños de las mismas y además se detallan las particularidades de cada una. Esto permite identificar los aspectos favorables y desfavorables que poseen cada una de los sistemas para de esta forma darle un seguimiento más detallado a su futuro perfeccionamiento.

3.1.1 Perfilado de datos PMMarc

El sistema de perfilado de datos PMMarc, propuesto por Borges Zamora (Borges Zamora 2016), es capaz de determinar los errores comunes en el módulo de Catalogación de la suite ABCD utilizando el estándar bibliográfico MARC 21, entre estos deslices se pueden destacar:

- validaciones técnicas,
- validaciones en base a reglas de negocio,
- análisis de los campos descriptivos,
- identifica campos comunes y estandarizados.

Como actor principal, el sistema presenta al catalogador, el cual puede identificar los errores existentes en sus catálogos, buscar en los registros los campos del nivel mínimo con valores ausentes, describir la cantidad de registros existentes para cada tipo de material, identificar la frecuencia de uso de los campos, determinar patrones erróneos en el año de publicación, entre otras funcionalidades.

La herramienta propuesta por Borges Zamora (2016), está compuesta por nueve clases principales, las cuales están desglosadas en tres paquetes, dispersos de la manera siguiente: uno para las clases de los componentes del visual, es decir, para las clases que se encargan de los componentes de la vista. Otro para las clases que trabajan con el estándar bibliográfico MARC 21 y un paquete para los reportes. (ver Figura 13)

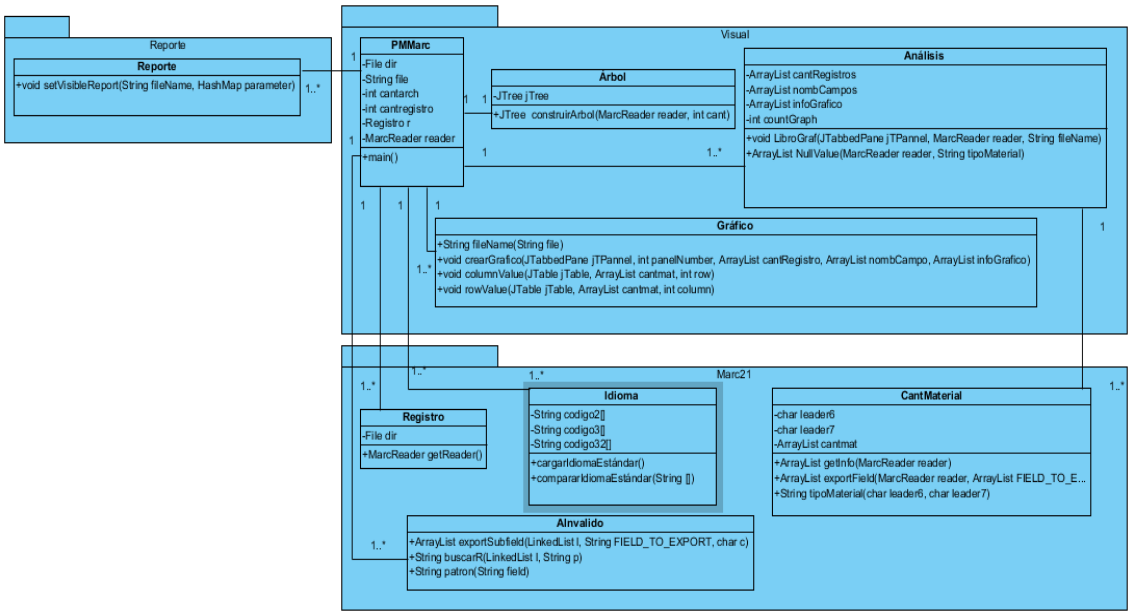

Figura 13. Diagrama de clases del sistema PMMarc. Fuente: (Borges Zamora 2016)

Dentro de las clases principales que permiten la visualización de los elementos en la herramienta, se puede referenciar a la clase “PMMarc”, la cual es encargada de manipular las vistas. La clase “Arbol” es la facultada para construir el árbol que visualiza la estructura de cada registro. Por otra parte, la clase Soporte es la encargada de escribir los valores en las tablas que visualiza “PMMarc” para una mejor comprensión de los resultados emitidos y “Analisis” examina los valores ausentes en los campos y crea los gráficos mostrados por la clase “PMMarc”.

La clase Reporte, que se encuentra en el paquete del mismo nombre, es la encargada de almacenar y mostrar los reportes de errores en los catálogos.

El grupo encargado de trabajar con el formato MARC 21 es el siguiente: La clase Registro es la que contiene las funciones para obtener la información almacenada en un archivo *.mrc. Las clases “Idioma”, “AInvalido” y “CantMaterial”, son las encargadas recoger los registros buscando los idiomas que no se corresponden con el estándar

bibliográfico MARC 21; determinar si es válido o no, el patrón de cada año de publicación y establecer la cantidad de registros por tipo de material, respectivamente.

Después de analizar, las principales características de PMMarc, propuesto por (Borges Zamora 2016), se destaca que esta se evidencia en la arquitectura propuesta, en la fase de análisis.

3.1.2 Medición y mejora de la completitud MARCCompleteness y MARComp

MARCCompleteness, propuesta por Guevara Torres (Guevara Torres 2016), para la medición de la completitud en todo el registro, cuenta con una interfaz sencilla que facilita el uso de la misma al actor del sistema. Esta herramienta permite gestionar colecciones de registros bibliográficos, realizar medidas de completitud mediante la implementación de métricas de esta dimensión y detecta duplicados existentes en estos registros. Dicha herramienta aborda las fases de definición y medición en la arquitectura propuesta.

Como una versión superior de la herramienta anterior, en (García Mendoza 2017), se propone el sistema MARComp, que además de medir la completitud, realiza la mejora de la dimensión.

En la Figura 14, se muestra el diagrama de clases correspondiente a medir la completitud en el sistema MARComp.

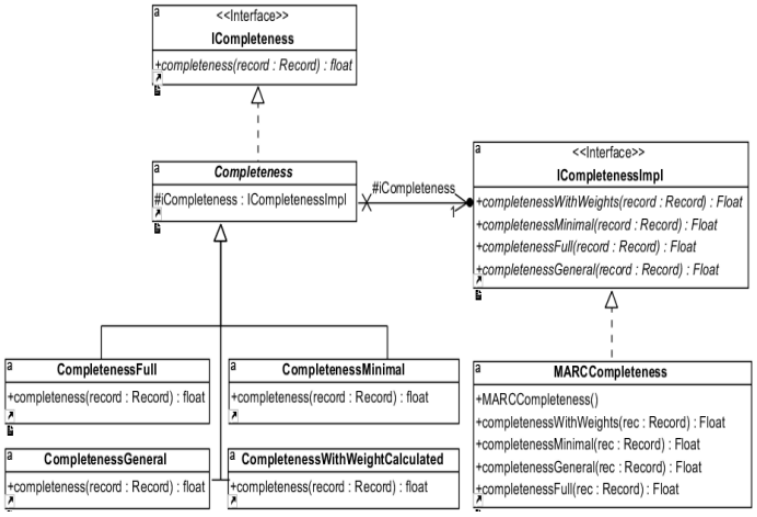

Figura 14. Diagrama de clases, asociado a medir la completitud en MARComp. Fuente: (García Mendoza 2017).

Se utiliza en MARComp, el patrón *Adapter*, debido a que las clases de la biblioteca MARC4J, que se utiliza para trabajar con el formato MARC 21, no contienen todos los elementos necesarios para representa un registro. El uso de este patrón se evidencia en la clase "*MARCRecord*", la cual contiene un atributo nombrado *record* que coincide con el tipo declarado en la interfaz *Record* de la biblioteca mencionada (ver Figura 15).

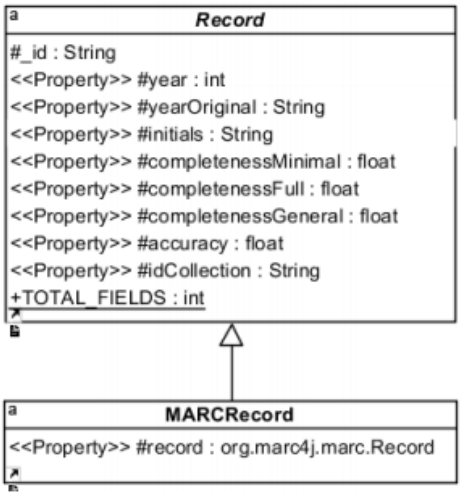

Figura 15. Clase MARCRecord, donde se emplea el patrón Adapter en MARComp. Fuente: (García Mendoza 2017)

La herramienta que se caracteriza, permite al catalogador, importar una colección determinada, medir la completitud en todo el registro en base a las diversas métricas definidas y realice la mejora de esta dimensión. Esta última está compuesta por la detección de registros duplicados y su desea o no mezclar los mismos. En caso de decidir mezclar los resultados obtenidos, es necesario ejecutar el primer aspecto antes mencionado de la mejora de la completitud. Como última opción del usuario del sistema, este tiene la opción de exportar los resultados de la colección de los registros en formato MARC 21.

Para una mejor organización, el sistema propuesto por (García Mendoza 2017), se encuentra estructurado en paquetes (ver Figura 16).

Figura 16. Diagrama de paquetes de MARComp. Fuente: (García Mendoza 2017)

La herramienta propuesta por (García Mendoza 2017), se evidencia en la arquitectura propuesta en las fases de definir, en la de medir y en la de mejora.

3.1.3 Medición y mejora de la exactitud en el campo autor MARCAccuracy

El sistema MARCAccuracy, propuesto por Lima Ortega (Lima Ortega 2016), para medir y mejorar en la dimensión exactitud, específicamente en el campo autor, presenta una interfaz sencilla y amigable que facilita su uso al usuario que interactúa con ella. MARCAccuracy es capaz de ejecutar medidas de exactitud a los registros

bibliográficos, a través la implementación de una métrica para la medición de esta dimensión.

Para una mejor organización de la arquitectura de la herramienta, MARCAccuracy, proporciona un grupo de clases (ver Figura 17) agrupadas en paquetes para una mayor organización de la misma (ver Figura 18). Además, utiliza la biblioteca MARC4J para el trabajo con registros MARC, la cual provee una interfaz para la lectura y escritura en este formato.

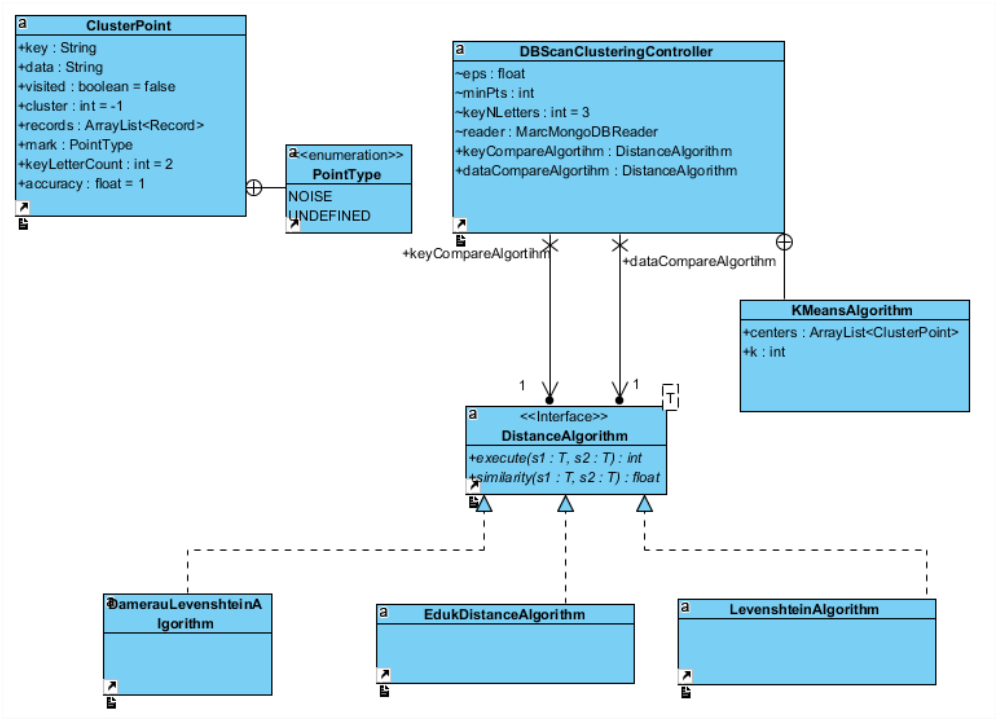

Figura 17. Diagrama de clases de la herramienta MARCAccuracy. Fuente: (Lima Ortega 2016)

Como solución computacional del sistema, MARCAccuracy define una entrada de ficheros en formato MARC 21, almacenando estos, para aplicarle la medición de la exactitud específicamente al campo autor, y la herramienta provee una salida en los mismos formatos de entrada.

En la arquitectura propuesta, la herramienta MARCAccyracy, se evidencia en las fases de Definir, Medir y Mejora.

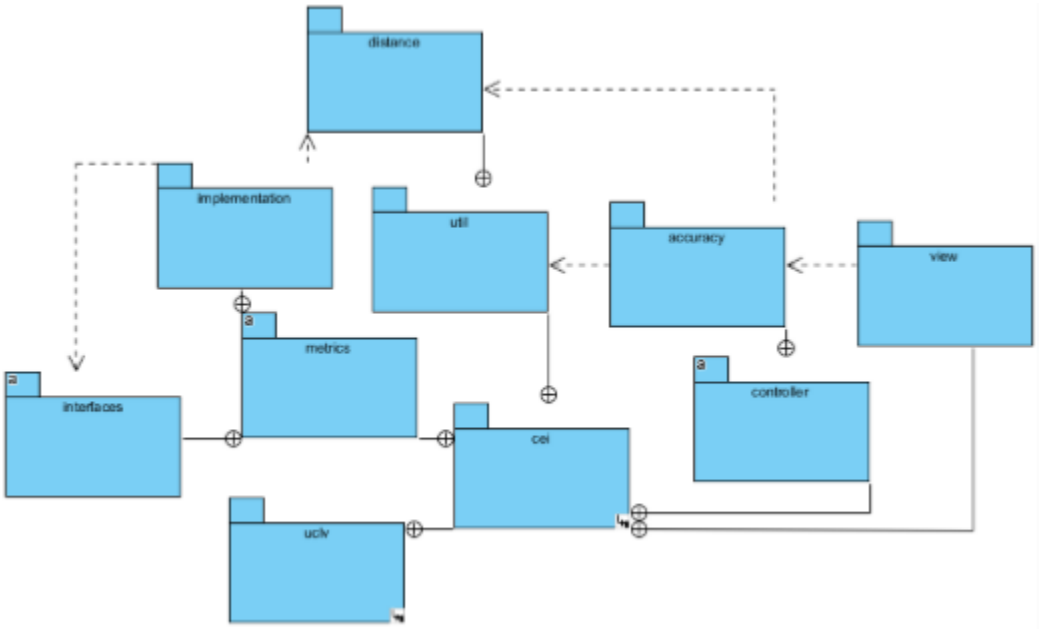

Figura 18. Diagrama de paquetes utilizado en MARCAccuracy. Fuente: (Lima Ortega 2016)

3.2 Adaptación de las herramientas a la arquitectura propuesta

Para adaptar las herramientas existentes a la arquitectura propuesta, se tienen en cuenta las características específicas de cada una. Realizado esto, se puede identificar que habría que modificar en la arquitectura de las herramientas existentes para que se integre al diseño propuesto.

En el sistema PMMarc propuesto por Borges Zamora (Borges Zamora 2016), es necesario adaptar la herramienta para incorporar las clases a la arquitectura propuesta. Se puede expresar disímiles de ideas con relación a que debería de hacerse para ajustar la arquitectura de PMMarc. La propuesta en la presente investigación, es ubicar los métodos relacionados con el perfilado de los datos en una clase dentro del paquete "*profiler"* y que, además esté relacionada con la clase *marc21Record*, ubicada en *record*. En el caso de los reportes, estos pueden incluirse en el propio paquete de perfilado o crear un nuevo paquete con este fin.

En el caso del sistema MARComp, propuesto por (García Mendoza 2017), ya contiene paquetes con un uso similar a la arquitectura propuesta, por lo que es más fácil su adaptación. Por ejemplo, ya contiene un paquete "*record"* que contiene la clase MARC21Record, contiene, además un paquete para las operaciones, que, aunque se encuentre dentro de otro, sus clases pueden ser utilizadas en la nueva arquitectura. En el

caso de las dimensiones, a pesar de que la estructura es "*metrics.completeness"*, también se pueden incluir las clases en la arquitectura propuesta.

MARCAccuracy, propuesto por Lima Ortega (Lima Ortega 2016), es un sistema que realiza la medición y la mejora de la calidad de un catálogo, concretamente en la dimensión exactitud en el campo autor. Es necesario adaptar la arquitectura que presenta el sistema que se describe, si se desea integrar. Para ello se toma propuesta de solución, se toma los métodos relacionados con la dimensión exactitud y ubicarlos en la ruta "*dimension.accuracy"*. Lo mismo sucede con las métricas correspondientes a la dimensión descrita, tomar los métodos asociados a esta, y ubicarlos en "*dimension.accuracy.metrics*", siempre un cuando estén relacionada con la clase *marc21Record*.

3.3 Propuesta de diseño de la herramienta integrada

Como diseño visual de la arquitectura propuesta, presenta una interfaz sencilla y amigable, donde el usuario final es el catalogador. En la ventana inicial del sistema (ver Figura 19), la misma cuenta con dos secciones principales.

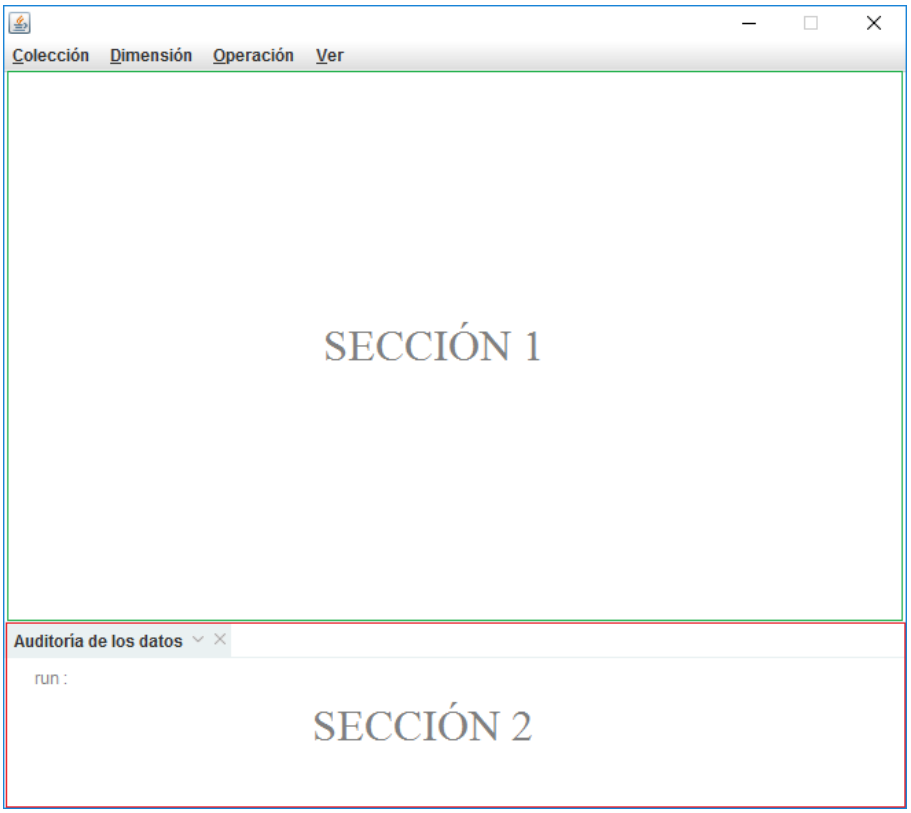

Figura 19. Ventana principal de la herramienta propuesta. Fuente: (Elaborado por el autor)

La sección 1 (representada en un cuadro de color verde), es el área asignada para visualizar las acciones a realizar sobre una dimensión específica, mientras que, la sección 2 (representada en un cuadro de color rojo) es el área establecida para la auditoria de los datos, es decir, cada acción que se realice sobre una colección determinada, se almacena en este espacio, para su posterior comparación con otros resultados emitidos y determinar, si mejora o empeora la solución.

Presenta una sección en el menú, asignada a las colecciones (ver Figura 20), la cual permite importar y exportar una colección determinada.

Figura 20. Lista de opciones que permite el menú Colección. Fuente: (Elaborado por el autor)

Al seleccionar la opción de "Importar", esta muestra una ventana modal (ver Figura 21), la cual permite seleccionar de qué destino deseas importar la colección. Estas pueden ser: desde un fichero, desde una base de datos SQL o desde una base de datos no SQL.

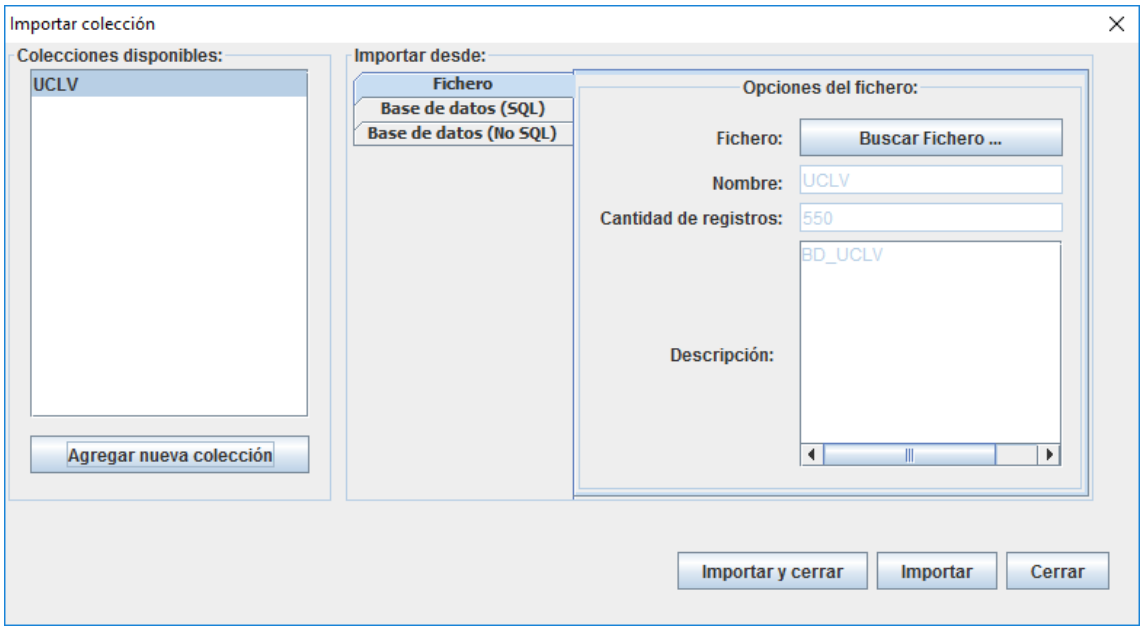

Figura 21. Ventana de importar coleccion. Fuente: (Elaborado por el autor)

Importar colección desde un fichero

Se selecciona el archivo MARC 21, el cual se representa a través de la extensión *.xml*, *.mrc* o *.json*. a través de un *FileChooser.*

Importar colección desde una base de datos SQL o base de datos no SQL

Se cargan por introspección las bases de datos correspondientes en un *ComboBox (ver* Figura 22*)*. Al seleccionar la base de datos correspondiente, se cargan en tiempo de ejecución los atributos que la misma posee para establecer la conexión.

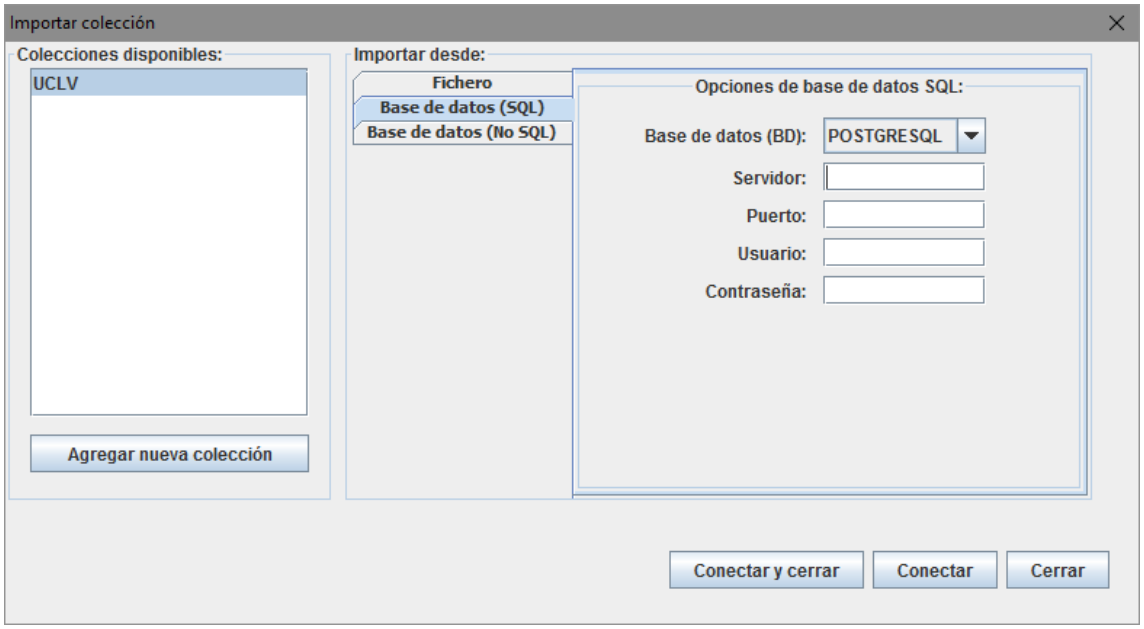

Figura 22. Ventana modal que permite importar una coleccion desde una base de datos SQL. Fuente: (Elaborado por el autor)

En la segunda opción del menú contextual (Dimensión), se listan las siete dimensiones que se plantean en (Bruce & Hillmann 2004) para medir la calidad (ver Figura 23).

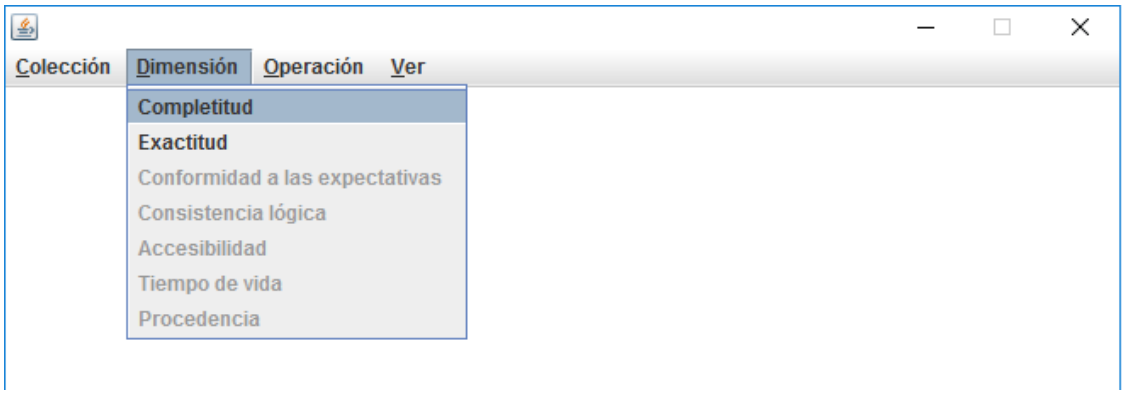

Figura 23. Lista de opciones del menú "Dimensión". Fuente: (Elaborado por el autor)

3.4 Conclusiones parciales

1. En este capítulo se caracterizaron las herramientas existentes que brindan soluciones aisladas a algunos de los problemas planteados por (Andreu Alvarez

2015). Se identificaron las fases que abarcan, en la arquitectura propuesta, cada una de las herramientas existentes. Se propuso como debían de adaptarse cada sistema en particular, para integrarse a la arquitectura propuesta y se diseñó un prototipo en *Java* de la interfaz visual de la herramienta integrada.

CONCLUSIONES

2. Se identificaron las fases que se ajustan a los metadatos bibliográficos a partir del análisis crítico realizado a varias metodologías de calidad de datos expuestas en la literatura.
3. Se diseñó una arquitectura que integra las fases identificadas.
4. Se diseñó un prototipo en *Java* de la interfaz visual de la herramienta integrada a partir del diseño anterior haciendo uso de patrones de diseño y el patrón MVC.

RECOMENDACIONES

1. Integrar las herramientas existentes, haciendo uso de la arquitectura y del diseño prototípico propuestos.
2. Incorporar las restantes dimensiones de calidad que plantean (Bruce & Hillmann 2004) para medir la calidad de los metadatos bibliográficos.
3. Incorporar otros estándares bibliográficos a la arquitectura propuesta.

REFERENCIAS BIBLIOGRÁFICAS

Andreu Alvarez, Y., 2015. Análisis de la calidad de datos en fuentes de la suite ABCD. , p.107.

Batini, C. et al., 2009. Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, 41(3), pp.1–52.

Batini, C. & Scannapieco, M., 2016. *Data and Information Quality. Dimensions, Principles and Techniques*, Springer International Publishing Switzerland.

Beall, J., 2006. Metadata and data quality problems in the digital library. *Journal of Digital Information*, 6(3), pp.1–20.

Bobrowski, M., Marré, M. & Yankelevich, D., 1999. Measuring data quality. *Universidad de Buenos Aires. Report*, pp.2–99.

Borges Zamora, M., 2016. Herramienta de perfilado de metadatos en formato MARC 21.

Bruce, T.R. & Hillmann, D.I., 2004. The continuum of metadata quality: defining, expressing, exploiting. In A. L. Association, ed. *Metadata in Practice*. ALA Editions.

Burbeck, S., 1992. Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC). *Smalltalk-80 v2*, 80(Mvc), pp.1–11.

Caplan, P., 1995. You call it corn, we call it syntax-independent metadata for document-like objects. *Public-Access Computer Systems Review*, 6(4), pp.19–23.

Deming, W.E., 1986. Out of the crisis, Massachusetts Institute of Technology. *Center for advanced engineering study, Cambridge, MA*, 510.

Dempsey, L. & Heery, R., 1998. Metadata: a current view of practice and issues. *Journal of documentation*, 54(2), pp.145–172.

Díaz-de-la-Paz, L. et al., 2015. *Calidad de datos*, Samuel Feijó.

Fall, N., 2005. Design Patterns 6.170. *Development*, pp.1–16.

Fernández, G. & Lenzo, N., 2010. Software ABCD Software ABCD (Automatizaci (Automatización de Bibliotecas n de Bibliotecas y Centros de Documentaci y Centros de Documentación). *1ra Jornada "Temas actuales en Bibliotecologia".Argentina*.

Gamma, E. et al., 1995. *Design Pattersns: Elements of Reusable Object-Oriented Software*,

García Mendoza, J.L., 2017. Procedimiento para la mejora de la completitud en registros bibliográficos con formato MARC 21.

García-Mendoza, J.L. et al., 2016. Herramienta CompMARC para la medición de la completitud de registros bibliográficos en formato MARC 21. *Revista Publicando*, 3(6), pp.397–407.

Grand, M., 2002. *Java Enterprise Design Patterns: Patterns in Java*, John Wiley & Sons.

Guevara Torres, P., 2016. *Medición de la completitud y detección de duplicados en metadatos con formato MARC 21*.

Juran, J.M. & Gryna, F.M., 1980. Quality planning and analysis.

Koronios, A., Lin, S. & Gao, J., 2005. A Data Quality Model for Asset Management in Engineering Organisations. *International Conference on Information Quality (MIT IQ Conference)*, p.25.

Krasner, G.E. & Pope, S.T., 1988. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal of object oriented programming*, 1(3), pp.26–49.

Larman, C., 1999. UML y patrones. , pp.1–265.

Ledesma, R., 2004. Sistemas estadísticos de propósitos múltiples: una revisión de programas gratuitos. *Metodología de Encuestas. Revista de la Sociedad Internacional de Profesionales de la Investigación en Encuestas*, 6(2), pp.105–117. Available at: http://casus.usal.es/pkp/index.php/MdE/article/view/956/897.

Lima Ortega, E., 2016. *Herramienta para la medición de la exactitud en el campo autor del formato MARC 21*.

Martínez Juan, F.J. & Cueva Lovelle, J. manuel, 1999. GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO. *Psicothema*, 11(003), pp.679–689.

Medrano, J.F., Figuerola, C.G. & Alonso, J.L., 2012. Repositorios Digitales en España y calidad de Metadatos. *Scire: representación y organización del conocimiento*, 18(2), pp.109–121.

Méndez Menéndez, J., 2014. Aplicación web escalable para gestión de información geolocalizada.

Moges, H.-T. et al., 2013. A multidimensional analysis of data quality for credit risk management: New insights and challenges. *Information & Management*, 50(1), pp.43–58.

Sadiq, S., Yeganeh, N.K. & Indulska, M., 2011. 20 Years of Data Quality Research : Themes , Trends and Synergies. *Proceedings Australian Database Confernece*,

Perth, Australia, 115, pp.1–10.

Salomone, S., Hyland, P. & Murphy, G.D., 2011. Perceptions of data quality dimensions and data roles.

Scannapieco, M. & Catarci, T., 2002. Data Quality under the Computer Science perspective. *Computer Engineering*, 2(2), pp.1–12.

Styles, R., Ayers, D. & Shabir, N., 2008. Semantic marc, MARC21 and the semantic web. *CEUR Workshop Proceedings*, 369.

Taylor, A.G., 2004. Organization of Information.

Tennant, R., 1998. Century cataloguing. *Library Journal*, 123(7), pp.30–31.

Wand, Y. & Wang, R.Y., 1996. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11), pp.86–95.

Wang, R.Y., 1998. A product perspective on total data quality management. *Communications of the ACM*, 41(2), pp.58–65.

Wang, R.Y. & Strong, D.M., 1996. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4), pp.5–33.

Xu, A., 1998. Metadata conversion and the Library OPAC. *The Serials Librarian*, 33(1-2), pp.179–198.

Yeganeh, N.K., Sadiq, S. & Sharaf, M.A., 2014. A framework for data quality aware query systems. *Information Systems*, 46, pp.24–44.