O'REILLY®

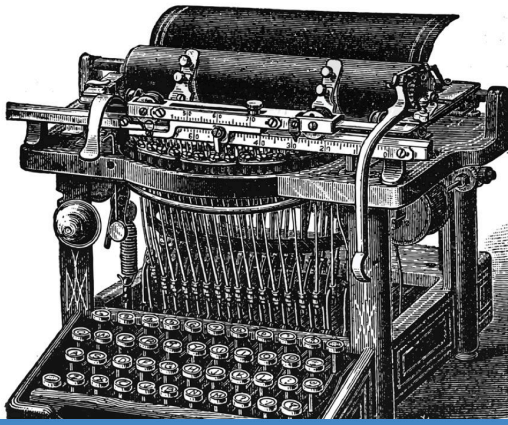
Compliments of TIBCO Jaspersoft®

Is Reporting Dead?

The Evolution of a Classic Information Delivery Strategy

Teodor Danciu
& Tom McKibben

Can your antique reporting tools run on Docker and report from noSQL?



Bring enterprise reporting to the 21st Century.

TIBCO®
Jaspersoft®

Is Reporting Dead?

The Evolution of a Classic Information Delivery Strategy

Teodor Danciu and Tom McKibben

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Is Reporting Dead?
by Teodor Danciu and Tom McKibben

Copyright © 2017 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Tim McGovern
Production Editor: Melanie Yarbrough
Copyeditor: Rachel Monaghan
Proofreader: Charles Roumeliotis

Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

March 2017: First Edition

Revision History for the First Edition
2017-03-01: First Release

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Is Reporting Dead?*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-491-97791-0

[LSI]

Table of Contents

Is Reporting Dead?

The world is made of data. Our human senses process many gigabytes of information every day, every moment of our lives. Much of this information is essential for our survival. Even so, the vast majority is lost shortly after it is collected. About 5,000 years ago, someone decided that it was a good idea to start recording important information and presenting it in a form that made it easy to make sense of and profit from that data. Thus the report was born!

As humanity grew and civilization spread across the globe, the need for reporting to understand and manage resources as well as promote commerce and science also grew. In particular, there was a need to generate reports in a timely fashion, which led to the use of mechanical calculating devices, later followed by electronic computers. One of the first high-level programming languages, COBOL, was specifically created with reporting purposes in mind.

Computing hardware and software continued to evolve at a rapid pace, which led to several paradigm shifts in reporting. New ways to access data developed due to the rise of the World Wide Web and the need to not only manage data on a petabyte scale, but also deliver it to billions of devices around the world. Reports are no longer just a static affair printed on paper, delivered once per quarter. Instead they have become dynamic, interactive, and deliverable on demand in real time. And simple charts and graphs have given way to sophisticated, interactive visualizations often designed by end users rather than professional analysts.

A recent survey showed that many of the largest corporations still get 88% of the data they rely upon from reporting. So, very far from being dead—even in an age of big data, data science, and machine

learning—reporting still has a big role to play. In this report we'll present a high-level overview of where reporting has been, where it is now, and where it may be going in the future.

The History of Reporting

One of the earliest examples of human writing is a Sumerian clay tablet known as the Kushim Tablet (see Figure 1-1). The symbols pressed in clay describe a transaction: "29,086 measures barley, 37 months. Kushim." The earliest human writing is a report! This, and similar ancient documents, reflected the need of an increasingly large and complex society to keep track of information important for its functioning and survival. The Babylonian census of 3800 BCE was very important for answering the question of how many people were in the population and how much food was required to feed them all.

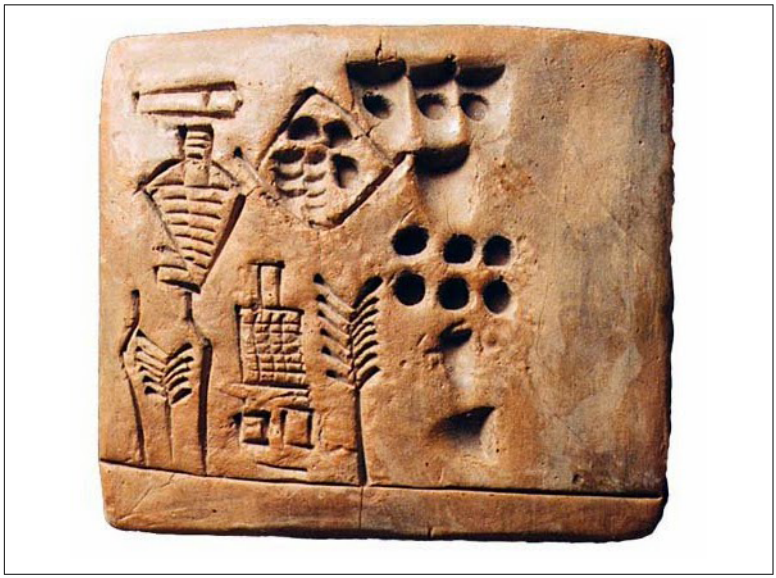


Figure 1-1. The Kushim Tablet

Reports answer important questions like: How much? How many? And how often? How much food is available? How many people need to be fed? And how often do these people eat? Needless to say, gathering that information and recording it on clay tablets was probably not the easiest thing to do. Imagine visiting all of the farmers in a vast region on foot, or sending messengers, and recording

their data; gathering all of this data at a central location, or perhaps several; and then manually analyzing and generating the final report. Producing such a report may have taken months of work. More than 5,000 years would pass before the often-arduous task of producing reports would be taken up by computers.

One of the first computerized reporting systems was implemented by the United States Census Bureau in 1951. It utilized the UNIVAC I computer to tabulate some of the 1954 economic census data. UNIVAC I was the first commercial computer produced in the United States. It weighed 13 metric tons and was built using 5,200 vacuum tubes.

Perhaps the first modern reporting system was the IBM 305 RAMAC in 1956. This was a massive machine, 30 feet by 50 feet. It was the first commercial computer to have a hard disk storage system. The disk drive was 16 square feet in area, weighed about a ton, and could store 5 MB of data. The system enabled businesses to create printed invoices. There was no programming language or user interface as we understand them today. Instead, physical wire jumpers were connected in sequence on a plugboard control panel to generate the desired output (see Figure 1-2).

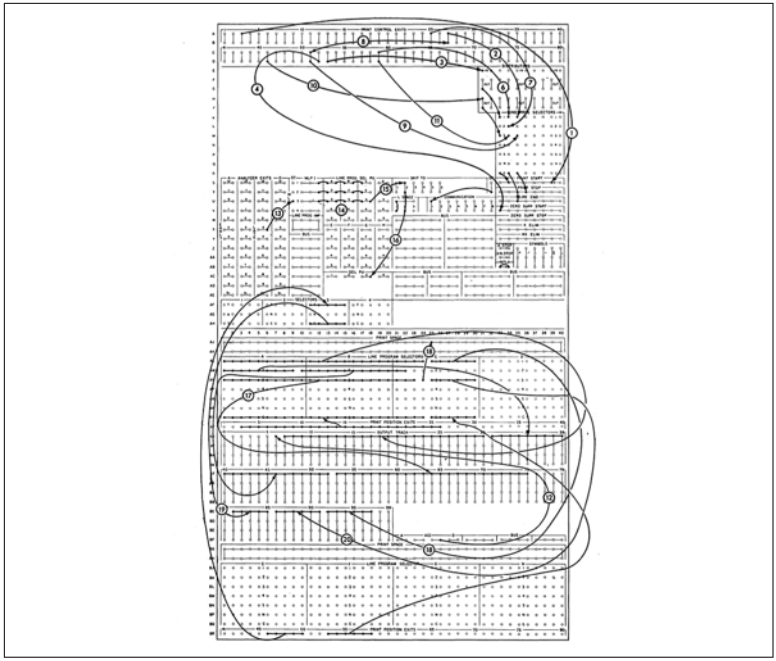


Figure 1-2. Plugboard wired to print multiple lines

Even after programming moved from hardware—physically connecting wires—to code, programming the production of reports would remain a daunting task until the introduction of COBOL in 1959. Based on FLOW-MATIC, a previous language invented by Grace M. Hopper, COBOL (Common Business-Oriented Language) is a high-level programming language designed specifically for common business-oriented operations. Its English-like syntax made it well suited to the creation of reports (see Figure 1-3). Now programmers needed only specify the report layout and the data to create a report. They no longer had to write code for things like page breaks and carriage returns.

```
REPORT HEADING
PAGE
CONTROL HEADING
DETAIL
CONTROL FOOTING
PAGE FOOTING
REPORT FOOTING
```

Figure 1-3. COBOL code for a simple report

The next big revolution in computing would take place in 1982 with the release of the IBM PC and the MS-DOS operating system. Bolstered by a strong marketing campaign, the IBM PC soon became a fixture in many businesses. And the operating system—which quickly became the standard on the PC clones that followed—sparked an explosion of software, including standalone reporting packages like Crystal Reports. This in turn led to an explosion of reporting. Many businesses now had access to the kind of computing power and software previously available only to large corporations and governments.

At the dawn of the 21st century, the reporting landscape was dominated by closed source proprietary applications. These were commonly integrated with business analytics and operational software. This landscape was about to change drastically. During the summer of 2001, software developer Teodor Danciu, coauthor of this report, was evaluating the feasibility of a large Java project that required the printing of complex report documents. He found that the available solutions were too expensive for the project's budget. But, having some experience working with reporting tools like Crystal Reports on the Windows operating system, he decided to build a Java-based reporting tool for the project himself.

While the project Danciu was evaluating did not ultimately get the green light, he decided to continue working on the reporting tool in his spare time on nights and weekends. He would go on to release the first version of the tool, JasperReports 0.1.5, on the open source software repository SourceForge in the fall of 2001. Danciu released JasperReports as open source software (OSS) in order to engage the

larger development community's assistance in enhancements and bug fixes of the reporting library.

After releasing JasperReports as OSS, Danciu found that he received a lot more input from the development community than initially anticipated. He had an initial set of features that he wanted to implement, but found that people were interested in using the tool to fill a number of other niches as well. Others were using the tool in more complex scenarios than he had imagined. Feedback of this kind revealed the need for a subreporting mechanism, requiring support for templates within templates.

Community code contributions to the project came in the form of patches that would fix bugs or add new features. The full-time JasperReports team would evaluate these patches, sometimes modifying them or rebuilding them to address unexpected issues that might not have been known by the contributor.

It was not at all certain in 2001 that open source would be the future of enterprise software, but the experience of JasperReports illustrates one of the greatest advantages of OSS: needed features can be immediately identified because the software is more widely available. And the availability of the source code itself allows others to implement features and fixes in a fully compatible manner. This also makes it easier to support other operating systems and build interfaces to other systems. And because OSS is not tied to any one company, it can survive and thrive even as companies come and go. This also makes it ideal for building infrastructure, which can be expensive and difficult to do in isolation. Much of the internet is built on OSS, observes Joe Drumgoole, MongoDB, Inc.'s Director of Developer Advocacy, EMEA: "Open source is now the default for infrastructure."

While closed source software from large companies like Microsoft or Oracle can be feature rich, the features are generally driven by only the largest market forces. So niche features may not be included in these products because there may not be any profit in developing them. Market considerations also pose a challenge for closed systems to keep pace with rapidly advancing technology. This becomes particularly problematic when security issues are discovered. OSS allows for faster discovery and remediation of security holes than is generally observed for closed, proprietary software.

JasperReports filled a significant need and, as open source software, rapidly became one of the most popular reporting tools in use thanks to a global community of developers. Over the last 10 years, JasperReports has seen over 493,000 active users and today averages over 15,000 downloads per month.

Reporting Versus Data Science and Data Discovery

There has never been more data available about more topics, from more sources, in the history of humanity. This data explosion has given rise to the discipline of *data science*, in which highly educated and trained data scientists use the tools of mathematics, statistics, and computer science to extract new knowledge and insights from the data. This practice is commonly referred to as *data mining*. It is a highly specialized process and not easily available or useable by most analysts.

Data discovery is the extraction of insights from data using tools that do not require science expertise (programming and statistics). These tools allow analysts and business stakeholders to explore and answer questions that come more directly from the concerns of their enterprises down to those of their individual departments and divisions.

Figure 1-4 breaks down the differences between reporting, data science, and data discovery.

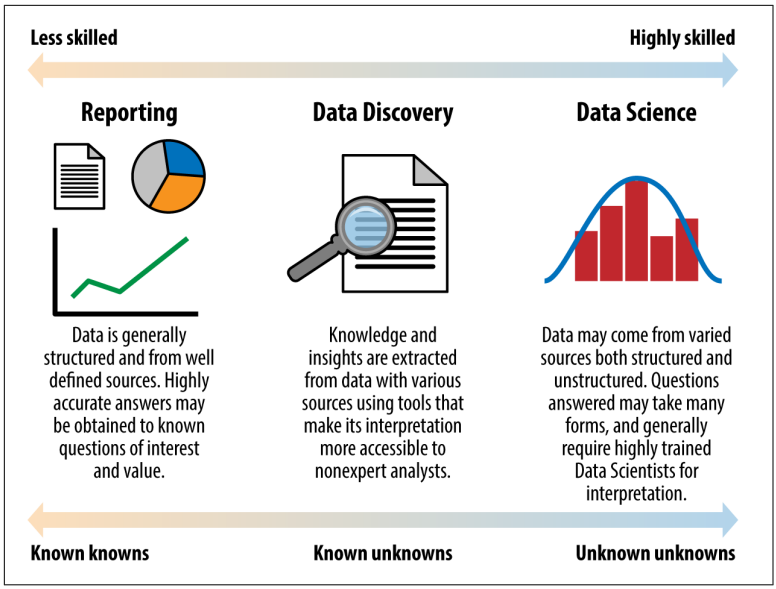


Figure 1-4. Comparison of reporting, data science, and data discovery

The availability of continually updating information has also made *streaming analytics*, the real-time analysis of data, possible. In an industrial setting, this might take the form of quality control sensors monitoring a manufacturing process and sending that data to a real-time dashboard application. This allows immediate action to be taken if the product goes out of tolerance. In a retail setting, real-time sales data from individual stores makes it possible to immediately identify popular products and the stores that need to restock them to avoid any loss of sales.

So where does reporting fit into this data landscape? Is it time to kick it to the curb and join the cool kids doing data discovery and streaming analytics? By no means! Reporting is still alive and kicking. As noted earlier, reporting answers questions like: How many? How much? How often? These are implemented as well-defined questions of proven importance to the stakeholders requesting the reports. Their data sources are also known and well defined, usually structured in nature. This allows reports to produce answers with a high degree of accuracy. That accuracy still matters, and reporting has evolved to meet the challenges and opportunities posed by the new world of data. Robert Zazueta, Global Director of Digital Strategy at TIBCO, notes, "The evolution has been towards more data,

faster data, and as a side effect, more real-time data. The kind of reports you used to expect from this were high-level executive briefings; that's what the end users used to get out of this. Most end users still want an executive-level report, but they want that executive-level report to reflect what is as close to now as possible."

Paradigm Shifts in Reporting

The rapid evolution of computing hardware and software has led to profound paradigm shifts in reporting. It is no longer dominated by the periodic generation of well-defined, static reports printed on paper, to be read by professional analysts. Today's reports are dynamic, generated from a multitude of data sources, and interactively formatted for consumption by everyone from web-conferenced boardroom executives to middle schoolers in the park with smartphones.

Shift 1: New Ways to Access Data

In the days prior to the rise of the internet, a reporting application running on a server would access data via a data access layer implemented by driver code. For an application written in Java, this would be via JDBC (Java Database Connectivity) drivers through which SQL queries would be made against the relational databases housing the required data. An end user would interact with this server application using a desktop client over a local area network in a client-server arrangement.

The arrival of the internet and the World Wide Web led to an explosion of data sources and web-based clients. This has required the implementation of new ways to access reporting data. To support clients running on desktop, mobile, and other devices, web service APIs have been implemented on servers generally written to either the SOAP or REST protocols, and data is typically passed in JSON format like the following description of an automobile:

```
{
  "make": "Limited Motors",
  "model": "Mountaineer",
  "year": 2016,
  "color": "Sunset Orange"
}
```

In using data from these web services for *reporting* purposes, there are two considerations to take into account. The first is how to parse or digest the returned data and format it into something that can be used by the reporting tool. In the case of JasperReports, a special query language was created to transform the data into virtual sets of records that one could iterate through and extract named values from (as with a result set returned by a SQL query). The second challenge is how to connect to remote data sources. Most of the time this means connecting to REST APIs over HTTP and often requires custom ways to authenticate to access data securely. JasperReports had to expand in this area considerably to meet these new requirements.

The nature of data stores has also shifted to accommodate new client demands for data that is more varied and has a different shape than the row-oriented tabular data housed in relational databases and returned by SQL queries. NoSQL databases are being employed to serve data that needs to be read fast and may not be rectangular in shape. That is, the data does not always have the same number of columns in each row. Popular NoSQL data stores include MongoDB, Cassandra, and Elastic (Figure 1-5).

Figure 1-5. Popular NoSQL data stores

Data volume has also increased by orders of magnitude beyond what the traditional row-oriented RDBMS (relational database management system) can handle in a timely fashion. It was, after all, the need to scale quickly that led to the creation of NoSQL databases. These had to be able to run across large clusters of computers where at any given time one or more of them might be inoperative. Fault tolerance like this cannot be handled by SQL-based databases scaled across a large cluster. NoSQL databases like MongoDB were relatively easy for developers to set up, and in the happy event that their web application became the next big thing, easy to scale up and meet demand.

Another approach to handling this aptly named *big data* was with data stores built around column-based datasets and the MapReduce

paradigm on Hadoop. Amazon Redshift is the prime example of commercializing this paradigm in the cloud (see Figure 1-6).

Figure 1-6. Popular big data databases

The combination of NoSQL and big data has required the development of an entirely new set of purpose-built tools, such as Hive and Pig, to fulfill the role once served by SQL queries.

Reporting tools have likewise had to further evolve to support the demands of big data. JasperReports has done this through the addition of special connectors and pluggable extensions. The arrival of the Internet of Things (IoT) will likely continue the growth of demand for big data–driven reporting. While data backends will always continue to evolve, humans will want answers from their reports, not data. So complexity increases on the backend while simplicity rules on the frontend.

Shift 2: From Static Outputs to Interactive Omni-Outputs

The earliest output of running a computer-produced report was something printed on paper (Figure 1-7). This output was defined well in advance of its final production and its format was rarely changed. Indeed, as we saw earlier, changing that output was a relatively arduous programming task on early mainframe-based systems. JasperReports grew out of a need to produce pixel-perfect report documents that looked exactly the same onscreen as they did on paper, and had a more user-accessible design process built around customizable templates. The PDF report document exemplifies this goal, but the output is a static product.

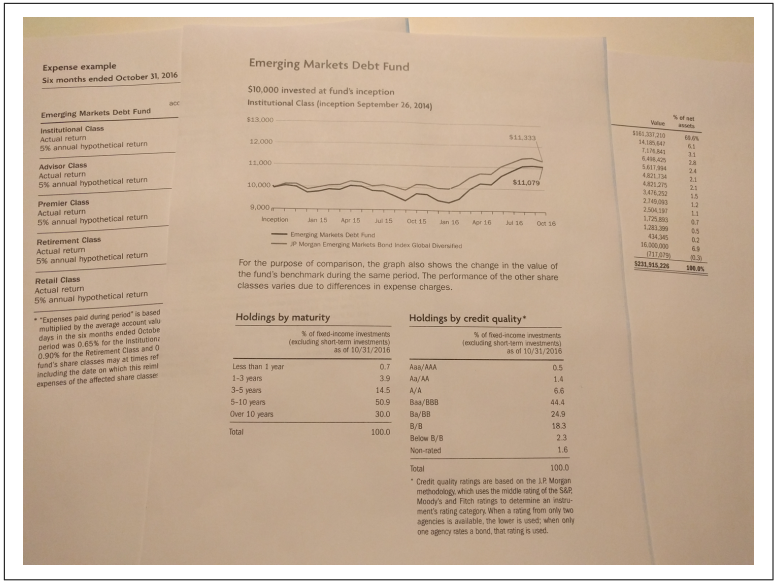


Figure 1-7. Classic printed reports

As computing hardware and software evolved and desktop clients developed more capabilities, end users wanted more interactivity with reports. Instead of just reading data in a static document, users want to be able to click on data in that report to open another document or view that might have additional details. In the same way, the user might also want to be able to navigate back and forth through collections of related reports and support documents.

Report generation libraries generally don't know how the parent application is using them. Any kind of interactivity with the report is the responsibility of the parent application—which might be anything from a Java Swing app to a rich web client running in the web view of a mobile application. Nevertheless, there has been a growing demand for support of such interactivity as sorting and filtering of data in the reporting tool itself.

As more developers were building web applications using the JasperReports libraries, Teodor Danciu and his team decided to build a framework that would allow people to build interactive report viewers within these applications. This framework allowed them to concentrate on adding the interactive features, such as sorting and styling on the fly, that people were asking for.

Now, feature-rich web and mobile clients allow interactive selection and formatting of reports on the fly, opening the door to new and sometimes unexpected insights into the data. And sometimes end user interactions are not as initially expected, either. Tom Kompare, a web developer at the University of Chicago Web Services, found this to be the case after releasing the Chicago Flu Shot Finder web app. When building the app he expected users to primarily make use of the search function to enter addresses or zip codes to find places where free flu shots were available. Instead, the analytics data showed that "people really loved the map. I was finding out they were just clicking on the map. So I made the maps bigger, and made the search a little less prominent on the screen." Kompare also made use of the responsive Bootstrap UI framework to accommodate differing screen sizes on mobile versus desktop clients.

Paper reports, like their clay tablet predecessors, offer zero interactivity. Now, thanks to the rapid evolution of report generation software, desktop reporting applications that allowed only limited interactions with the data have evolved into fully interactive web apps. And end user expectations continue to move in the direction of more options for interactivity.

Shift 3: From Periodical to Real-Time Delivery Methods

When generating a report required a lot of computation resources and time, it was generally run as a periodical batch job as often as the report was required. So a quarterly sales report would be run in a batch overnight once per quarter, with the printed output delivered to anyone needing a copy. A report needing wider distribution, such as a shareholder's report, might be sent to a print shop for mass copies that would be delivered by mail to its audience.

Rob Zazueta, Director of Digital Strategy at TIBCO Inc., reflected in an interview on his career in tech and data and the shifts that have changed the way we work with data. He's observed the shift from periodical to real-time reporting firsthand, and points out that APIs and other access tools were the catalyst that enabled the explosion in real-time data reporting.

At the start of his career, it was easy to describe the process. Managers wanted reports with very clear parameters. "The early workflow," he says, "was my boss would come to me and say, 'Hey, Rob Z, I need to find out how many people are using our website. I need to

find out how well this particular campaign is doing.' They'd give me the high-level parameters for what they wanted and then I'd have to figure it out based on the data I had in my databases and my logs." This process was slow, especially for ad hoc reports. "I had to do a lot of custom scripting. And I had to go through iterations back and forth between the person who requested it to make sure I was getting it right."

A flourishing of tools and APIs made it easier to get data—allowing the people who needed to understand the information to do the digging to create their own reports. Zazueta says, "Modern self-service tools allow me to go ahead and start digging in to data and figuring out the best way to find the information I'm looking for, and to visualize and present that information. It's not the technical nerds like me who are writing programs and writing scripts and trying to figure stuff out; it's now people who actually intimately understand the data and its context that create the visualizations and know to tag all of this appropriately so they can present it to the end user."

This gives more and more power to the end user, he says. "Once they've done it once or twice, the end user gets it. But these reporting tools also allow the end user to have some level of modification as well. They can drill in. They can try to figure out and create some of their own reporting tools—their own reports, I should say—because these reporting tools have made it easier for them to go ahead and do that."

This loop causes people to *want* to dive deeper into questions—meaning they are getting closer to the source of the data and requiring more detail. This means they are fetching data more often and making data-driven decisions more often. "The executive summary is showing me that here's how well we're doing right now on this particular campaign. Well, let's dive deep and see, compare that to other campaigns. Let's segment that according to user. Let's see how the campaign's doing on the East Coast compared to the West Coast. How it's doing for folks on mobile, for folks on the web?" These are all questions that modern interactive reporting makes it possible to answer.

Today, except for those with the largest datasets, most reports can be generated during the business day on demand. This has been made possible through the use of APIs that allow client programs to push data updates to servers over the network, as well as pull down query

result sets for use in reports. JasperReports has met this challenge through the creation of an API that facilitates its use in microservices that can be run across many computing nodes. This allows the scalability needed to handle heavy demand for data.

Shift 4: Visualization Types

Simple ASCII text graphics were the rule for the earliest reports, used to display line charts, bar graphs, and scatterplots on printed output (see Figure 1-8).

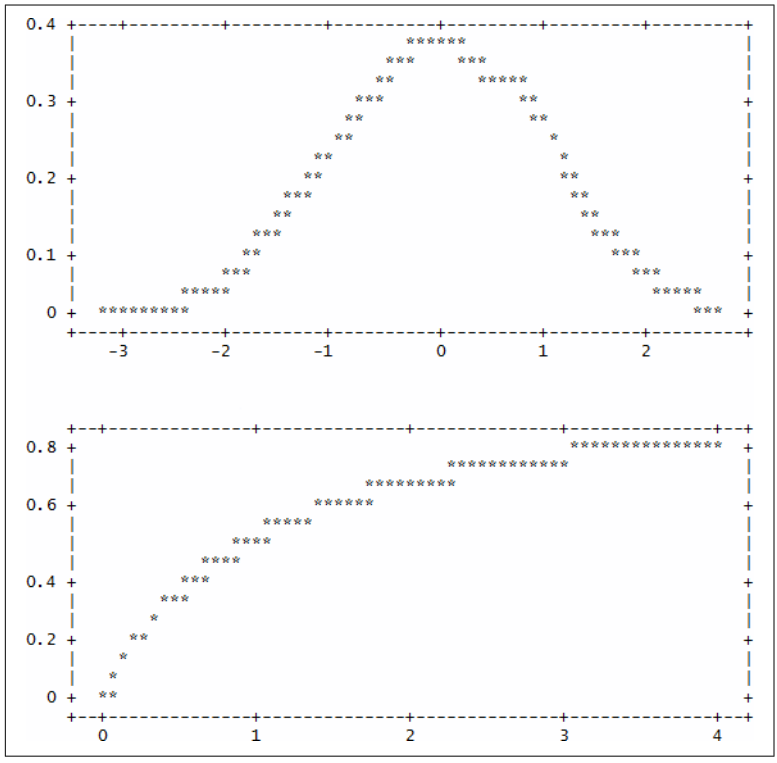


Figure 1-8. Plots rendered using ASCII graphics

Improvements in computing hardware would lead to much better-looking visualizations and open the door to new ones that were not possible previously. It was also the norm that the reporting program handled all of the output content, leaving little room for customization without requiring a software developer fairly early on or additional licensing fees for greater functionality.

The interoperability that the open source movement fostered accustomed users to using a range of tools in whatever combinations were required for a given use case. Developers responded to this demand in multiple ways. JasperReports used an open source Java charting package called JFreeChart that had a variety of chart types available that were a perfect fit.

The earliest examples of interactive web visualization solutions were based on Adobe Flash, which JasperReports supported by updating its API and adding support for pluggable components. This allowed for the creation of web report charts with a variety of animations. While Flash has seen its fortunes rise and fall, the pluggable nature of the JasperReports API has allowed it to keep pace and support new visualization tools built around HTML 5, CSS, and JavaScript. No Java programming necessary!

JavaScript frameworks such as D3 are making any type of visualization possible (Figure 1-9). They are also introducing interactivity into visualizations for which the computer display is the primary home. Virtual reality and augmented reality promise to literally add a new dimension to the display of data, which might even include audio representation. And the introduction of 3D printers is also adding a new possibility and dimension for the visual and tactile output of data.

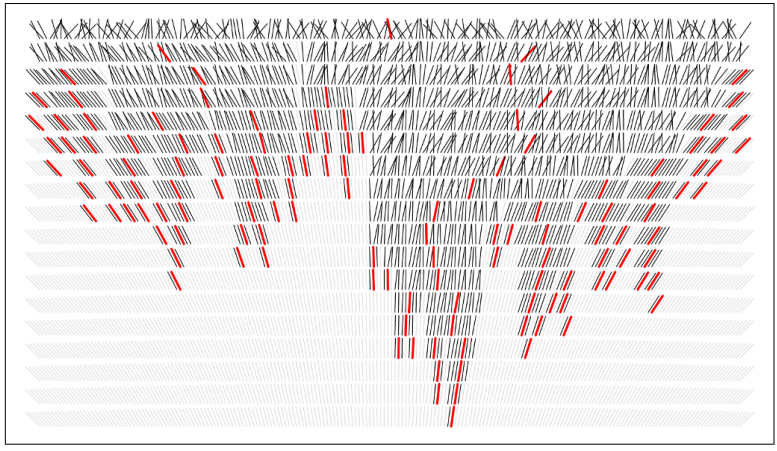


Figure 1-9. D3 has no limits as to types of data or concepts it can visualize; pictured here is the visualization of a sorting algorithm (source)

These new visualization options also largely exist independently of the reporting tool itself. As long as the data output is in a compatible format, such as JSON or XML, report designers—or, increasingly, end users—are free to customize how data is visualized using the package of their choice.

Reporting and Data Delivery in the Future

The creation of the web, and the rapid proliferation of mobile, connected devices, and people using them at home and on the job, has resulted in unprecedented data creation and consumption. New sources of data continue to be created along with new applications to consume that data. Systems originally designed to produce printed reports and analytics on a quarterly basis must now do so daily, or even hourly. And to make things worse, vendors are beginning to retire aging reporting tools and are replacing them with tools that cost more and don't always deliver the kind of reporting that many customers still need.

The need for high-frequency reporting (real-time dashboards and more) has led to the disappearance of the printed report. While PDF remains a popular format for pixel-perfect reports, reports are increasingly being displayed and consumed in other digital guises. And with that, the *business intelligence* (BI) that previously lived mostly on the server side now finds itself embedded in the report itself. Embedded BI is no longer bound to produce static outputs. Being closer to end users, it's evolving to respond directly to their requests in real time.

Reporting is evolving to take place closer to where the data is created and where decisions based on that data are made. Previously, all data was stored in a central location, and someone like a company vice president would use BI to inform decisions to be implemented in the coming quarter. Now, the data lives in many places, and someone like a store manager is using BI running in a tablet reporting app to decide how to set product prices today. Moving reporting closer to where it is actually used gives the information a context that ultimately leads to better and faster decisions. In the competitive retail markets of today, better pricing decisions are a matter of corporate survival.

As Rob Zazueta puts it, "There's going to be some level of automation behind it. The stuff that's going on with machine learning and

artificial intelligence really signifies a potential sea change in how we handle reporting. It will become a situation where the end user's going to say, 'I need answers to these questions,' and then you pass it on to the system. And the system, based on its AI machine learning, will identify the right sources, the right tags, and potentially do it more efficiently, potentially more quickly, and then be able to deliver that back to the end user. I think that's where we're going with all of this."

The movement of reporting closer to the end user would not be possible without the current networking and data infrastructure, which is largely built on open source software. The open, standard protocols built on OSS ensure that data is quickly and consistently delivered between systems running on a variety of platforms. And those open protocols themselves can be rapidly updated to adapt to the introduction of new devices and use cases.

The "Choose Your Weapons" Checklist

Reporting tools remain an indispensable part of making sense out of the torrent of data available for the end user. The most useful ones should satisfy most of the following criteria.

Data Flexibility

Reporting tools should work flexibly with data. This is particularly important given the growing diversity of data sources and formats. Look for:

- JDBC/ODBC support
- NoSQL drivers that allow query of data in their native language
- Flexible big data support that doesn't force use of "in-memory" analytics
- Caching layers to speed up report generation
- An extensible data layer that allows development of custom data adapters

Useable APIs

Supporting clients ranging from low-powered mobile and IoT devices to high-performance workstations requires easy-to-use APIs. Forward-thinking REST APIs also enhance data flexibility. Look for:

- Synchronous and asynchronous report generation
- Scheduling APIs

Visualization Support

Reporting tools should include support for the popular visualization frameworks being used to provide new insights into the data. Look for:

- Many built-in choices for visualizations
- Customizable chart options via API
- Support of third-party visualizations through libraries like D3

Embeddable and Extensible

Tools available as embeddable libraries give developers the freedom to craft custom reporting solutions. And an extensible architecture provides some future proofing and the ability to fill gaps. Look for:

- JavaScript embeddable APIs for putting reports into web apps
- Pluggable, extensible SDKs
- Flexible platform support for any OS and deployment options like Amazon and Docker
- DevOps-friendly build and deploy options

Open Source

Using open source tools backed by an engaged community is the best way to avoid vendor lock-in and ensure ongoing support and development. Open source is the best way to build custom solutions employing the tools most appropriate to the mission goals. Look for:

- Active and mature user and development communities
- GPL-licensed tools that are easy to extend
- Commercial support options

Licensed and Built for Scale

Just as reporting has escaped the boardroom, licensing must do likewise and scale to audiences of users from thousands to millions. Look for:

- Pricing options that are not constrained per user
- Horizontal scaling options through load balancing

From Kushim's Tablet to today, reporting is alive and well. Reporting tools have rapidly evolved, making reports more valuable to more people than ever before. End users now have the power to mold the clay and answer questions Kushim could only dream of.

About the Authors

Teodor Danciu is the founder and architect of the JasperReports library—the most popular open source reporting tool—and is now working for Jaspersoft. Before starting the JasperReports project in 2001, Teodor worked for almost nine years with several French IT companies as a software engineer and team leader on ERP and other medium-to-large database-related enterprise applications using mainly Java technologies and the J2EE platform. Teodor has a degree in computer science from the Academy of Economic Studies in Bucharest.

Thomas McKibben is a software engineer living in Chicago's Hyde Park neighborhood. After completing a PhD in Physics studying bottom quark decays at Fermilab, he spent some time programming for a market data company before moving into bioinformatics and consulting. He currently writes backend Java applications and iOS apps in Swift.