Arnaud Hebert
Promotion IRM 2012
Software Engineer Intern

ARECIBO

From conception to production: data visualization software application

3rd year internship report

September 2012

- Business-to-Business (B2B)
- Cloud-Computing
- Management Software
- IT Services
- User Interface
- Data Visualization
- HTML5, CSS3, JavaScript
- Design, Implementation
- Testing, Release

ESIL Referent: Léon Mugwaneza

Ariba Referent: Jacques Boutin

Acknowledgment

First, I would like to thank Jean-Philippe Préfot, who interviewed me in December 2011, and my supervisor, Jacques Boutin, that made me a part of the team for this very interesting project.

Then, I thank my team partners Olivier Francome, Elmehdi Assir and Slade Capasso with whom I worked with during this internship.

And I also thank Kingsley Ngan, another team partner, with his knowledge and expertise on user interface and his advice that helped me several times solving issues. I thank the project leader Dinesh Shahane with his very impressive skills in project architecture, project leading and computer science.

Finally, I thank all Ariba employees that I met during those six months. They were very nice and helped me feel comfortable at work.

Abstract

This abstract is a preview of my computer science engineering internship, that occurred from February to August 2012 in Sunnyvale, California. The project team is composed of four interns that came and left at different times, and Jacques, manager, Dinesh, project leader, Kingsley, project supervisor and Pallavi, program manager. Each one of us has different assigned tasks, but because of the dependencies of the application, we worked together.

The subject of this internship is building a data visualization application from scratch using data from the Ariba Network. The technologies used were HTML5, CSS3, and JavaScript on the User Interface front-end side and Java on the back-end side.

The tools used were Jetty, Tomcat and Virgo servers. We switched from one server to another because of dependencies. The dependencies are JSON libraries for the format of the communication message between the server and client that are implemented with WebSocket libraries, Quartz libraries for scheduled jobs, Servlet libraries to allow client to access server, HBase libraries to get historical data from the data base, and JavaScript libraries.

All the application architecture and these tools will be described later in this report. Also, the tools for document management such as Perforce, our Wiki and the Blog will be detailed.

Despite the Functional Requirements Document was not written until August, we started filling out the Design Specifications Document in order to deploy the Arecibo project as an Ariba project. Scrum was used to as a method for managing this project. Security and privacy reviews concerning data and visualizations needed to be done before we deploy it. This project will be deployed on TV screens in the office, and will not be available for Ariba customers yet.

Unfortunately, we only saw the application as a prototype and the deployment will be completed soon. The application had twelve visualizations, two based on historical data and the rest based on real-time data. Several meetings happened between the project team, teams that had data and the Ariba leaders. Our presentation skills improved and overall, they were proud of our work.

Table of Contents

I- First part: enterprise presentation

A- Identity and organization

1. Presentation

The enterprise where I have had my 3rd year internship is called "Ariba Inc". It has about 2400 employees stationed all over the world.

Its headquarters is in Sunnyvale (California) and the human resources office is located in Pittsburg (Pennsylvania).

2. Ariba, Inc's history

Ariba was founded in 1996 by seven collaborators.

- Bobby Lent
- Boris Putanec
- Paul Touw
- Rob Desantis
- Ed Kinsey
- Paul Hegarty
- Keith Krach

They believed on the idea of using the Internet to enable companies to facilitate and improve the procurement process. Procurement had been a paper-based, labor-intensive, and inefficient process for large corporations. According to the company's website, Ariba provides "Spend Management solutions" which help companies "analyze, understand, and manage their corporate spending to achieve cost savings and business process efficiency." Currently, 94 of the Fortune 100 and more than 200,000 other companies use Ariba's SaaS (Software as a Service) solutions to manage their spend and commerce activities.

Ariba was one of the first business-to-business Internet companies to go public (in 1999). The company's stock more than tripled from the offering price on opening day, making the three year-old company worth $6 billion. In 2000, the stock value continued to climb, and Ariba's market capitalization was as high as $40 billion. With the bursting of the dot-com bubble, Ariba's stock price fell dramatically to the low double digits in July 2001, where it has remained since, with a market capitalization of just over $1.5 billion as of June 2010.

3. Activities

Ariba, Inc. is the leading provider of collaborative business commerce solutions. Ariba combines industry-leading technology with the world's largest web-based trading community to help companies discover, connect, and collaborate with a global network of partners - all in a cloud-based environment. Using the Ariba Commerce Cloud, businesses of all sizes can buy, manage cash, and sell more efficiently and effectively. More than 730,000 companies around the globe use the Ariba Commerce Cloud to simplify inter-enterprise commerce and enhance results.

4. Ariba Solutions and Services for Successful Business Commerce

Companies of all sizes, and across all industries, need to drive maximum value from their business commerce initiatives. But commerce is hard. And technology alone won't make it easier. Successful business commerce, including spend management, supplier management, and working capital management, is driven by a combination of technology, service capabilities, and community resources. Working with Ariba solutions and services, you can successfully plan, deploy, and adopt an electronic business commerce program to deliver world-class results to your organization.

There are multiple challenges to getting the most out of your electronic business commerce initiatives, including:

- Developing a business case and program approach
- Setting expectations and securing executive support
- Fueling change management
- Marketing efforts internally
- Enabling trading partners
- Driving adoption by category, geography, function, etc.
- Testing technology for acceptance, data, and integration
- Accessing external resources and expertise to fill internal gaps

Companies must not only prepare for deployment of their technology, but the change necessary to fully leverage their new solution over the long haul. Ariba's technology and service capabilities are uniquely designed to help organizations do this. The tailored solutions combine the planning, deployment, trading partner collaboration, and continued adoption services necessary to optimize your technology investments and realize the efficiencies that electronic business commerce initiatives can deliver.

5. Ariba's Global Business Commerce Support

Ariba has provided spend management, supplier management, and working capital management support to companies around the world for more than a decade. And the most successful organizations share a few things in common:

- A comprehensive approach to electronic business commerce that incorporates technology, community, and capabilities
- A structured approach based on industry best practices
- A solid business case and ROI model that incorporates milestones and tracking devices

- Based on these characteristics, Ariba delivers all the resources and expertise that companies need to overcome the challenges to electronic business commerce, including:

- Change management strategies and templates
- Organizational maps and best-practice approaches
- World's-largest network of pre-enabled trading partners
- Flight-planning expertise by category, geography, and function
- Technology deployment and integration capabilities

With Ariba solutions, delivered through the Ariba Commerce Cloud, you can focus on what matters most-driving the maximum value to your bottom line and creating lasting enablement for your organization-and leave the rest to us. When it comes to electronic business commerce, no other provider has the experience, scale, and flexibility of Ariba.

6. AribaWeb

On February 19, 2009 Ariba announced AribaWeb, an open source framework for Rich Internet Applications. It is designed to generate a user interface automatically from base Java or Groovy classes and includes Object-Relational Mapping features. It also encapsulates AJAX functionality and has a broad selection of UI widgets.

B- Environment

1. Figures

Type	Public
Traded as	NASDAQ: ARBA
Industry	Internet Software & Services
Founded	1996
Founder(s)	Bobby Lent, Boris Putanec, Paul Touw, Rob DeSantis, Ed Kinsey, Paul Heggarty, Keith Krach
Headquarters	Sunnyvale, California, U.S.
Area served	World Wide
Key people	Robert M. Calderoni, Keith Krach
Products	Spend Management Software, Contract Management Software, And Financial Solutions.
Revenue	US$339M (FY 2009)
Operating income	US$15.5M (FY 2009)[1]
Net income	US$8.19M (FY 2009)
Total assets	US$668M (FY 2009)
Total equity	US$434M (FY 2009)
Employees	2,400
Website	www.ariba.com
730,000	Suppliers
	Buyers
	Transactions per
	$ exchanged per

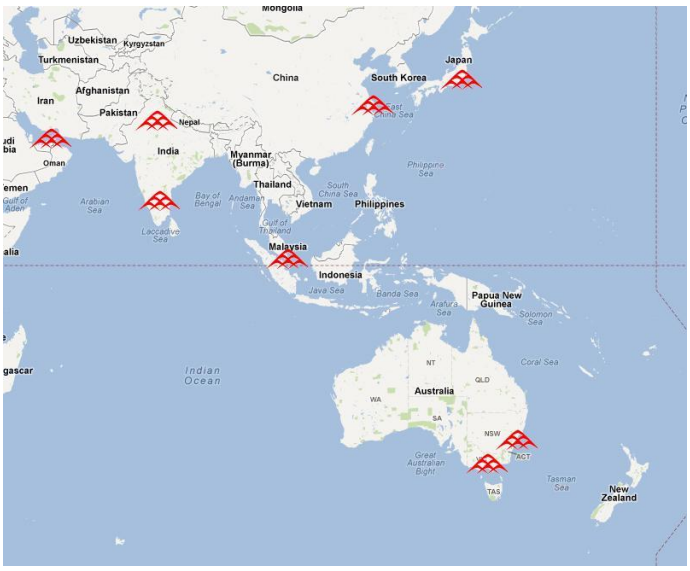
2. Offices

- North America:
  - Atlanta
  - Canada (Mississauga)
  - Chicago
  - Dallas
  - Detroit
  - Minneapolis
  - Pittsburgh
  - Sunnyvale (Headquarters)

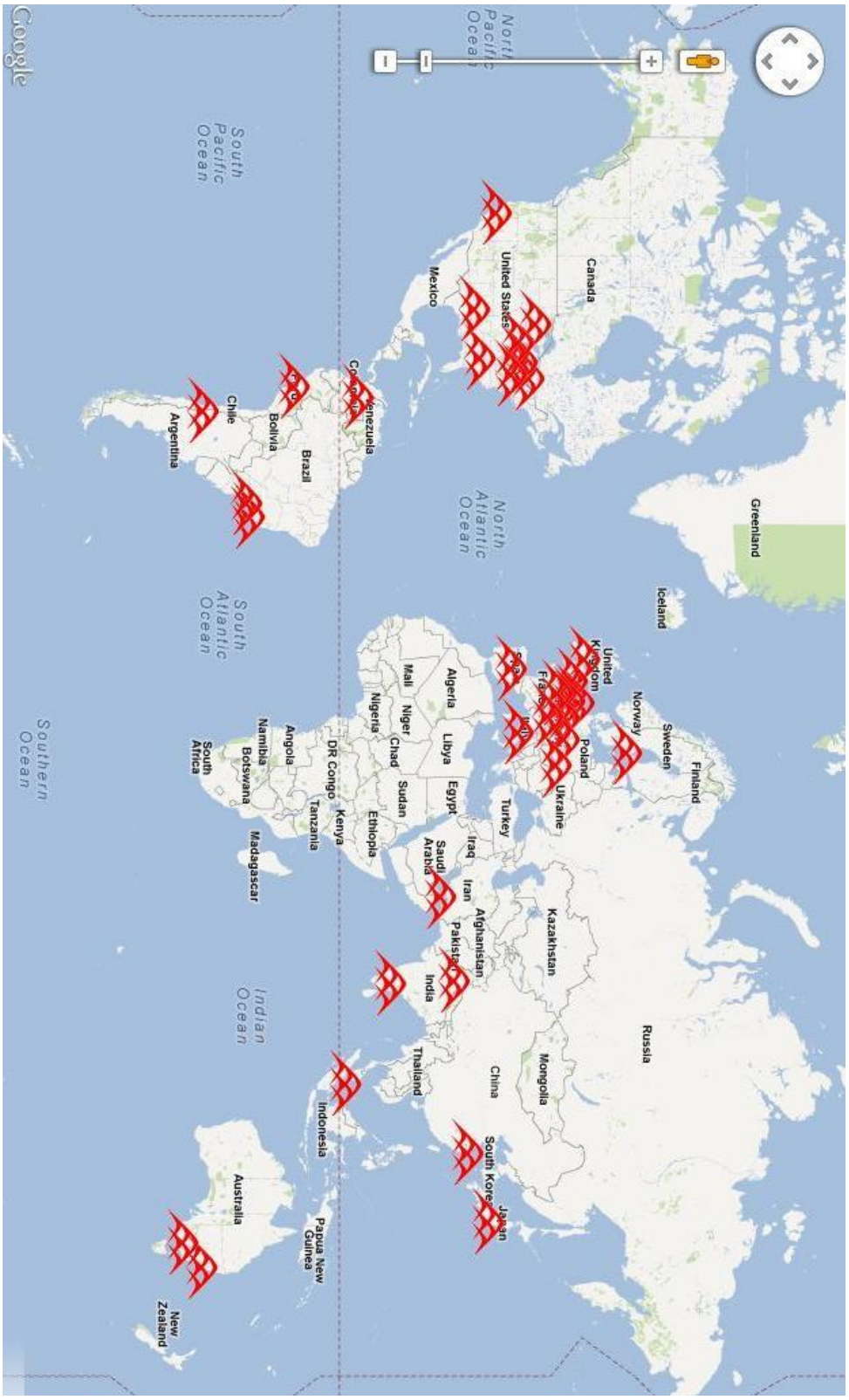
- Latin America
  - Brazil (Rio de Janeiro)
  - Brazil (Sao Paulo)
  - Columbia (Bogota)
  - Peru (Lima)
  - Chile (Santiago)

- Europe
  - Belgium (Heverlee)
  - Czech (Prague)
  - Deutschland (Frankfurt)
  - France (Paris)
  - Ireland (Dublin)
  - Italia (Roma)
  - Netherlands (Amsterdam)
  - Slovak Republic (Kosice)
  - Spain (Madrid)
  - Sweden (Stockholm)
  - Switzerland (Zurich)
  - UK (Surrey)

- Asia
  - Australia (Melbourne)
  - Australia (Sydney)
  - China (Shanghai)
  - India (Bangalore)
  - India (New Delhi)
  - Japan (Tokyo)
  - Singapore



In total, there are 32 offices on four continents (Asia, Oceania, America and Europe). There is no office in Africa but there are many customers in South Africa.



3. Ariba's Partners

Business commerce management aligns organizations, processes, and systems to drive best-value performance whenever enterprises large or small work together to buy, sell, or manage cash. Ariba is the world's business commerce network. Ariba combines industry-leading cloud-based applications with the world's largest web-based trading community to help companies discover and collaborate with a global network of partners.

Ariba's partners play a critical role in helping our customers understand and overcome the complex challenges associated with the enterprise software marketplace. Ariba's partners significantly enhance Ariba's ability to deliver collaborative, flexible software solutions.

Their strategy is designed to support our corporate growth objectives:

- Increase cloud-based, software-as-a-service subscription sales
- Drive and expand our Ariba Network solutions offerings
- Expand opportunity in underserved markets
- Increase our market coverage.

To attract, empower, and support world-class organizations that would collaborate with Ariba to achieve these objectives, we have developed a variety of programs with reciprocal partner benefits, including a new, exciting program centered around Ariba's market-leading sourcing solution.

Ariba's main partners are:

- Axys Consultants
- American Express
- Dell Boomi
- Hewlett-Packard
- Hitachi Software Engineering Co. Ltd.
- IBM Technologies
- Intel
- Microsoft
- Oracle
- Verisign

4. Ariba's Customers

Ariba's customers are successful companies of all sizes, operating in all industries, and located across all regions of the globe. Ariba is the leading provider of collaborative business commerce solutions because our customers continually challenge us to meet the needs of their growing and evolving businesses. And Ariba rises to that challenge, delivering flexible and scalable solutions and services that drive results.

Ariba's main customers are:

- Air Liquide
- British Airways
- Chevron
- Dell Boomi
- Hewlett-Packard
- Lexmark
- Target
- Tupperware

C- Acquisitions

1. Past acquisitions and competitors

On December 17, 1999 Ariba announced it would acquire Atlanta based TRADEX Technologies Inc. in a stock swap valued then at $1.87 billion. TRADEX was the leader in the nascent Digital Marketplace Software field. The stock market liked the acquisition and the price of Ariba's shares rose from $57 at the time of the announcement to $173 at closing on March 9, 2000, which also marked the peak of the Internet Bubble. The 33.2 million shares that Ariba issued to buy TRADEX were then worth $5.6 billion to TRADEX shareholders.

In January 2001 Ariba announced that it would acquire Agile Software in a $2.55 billion stock swap. By April, with Ariba facing a disappointing second quarter and cutting a third of its workforce, the deal had fallen apart.

In early 2004, Ariba acquired FreeMarkets which gave the company a software package both in the upstream (sourcing) and downstream (buyer) of the procurement process. In late 2007, Ariba took over the company Procuri, which enhanced the company's client base and on-demand abilities.

Ariba's competitors include PROACTIS, SAP, Bravosolution, Oracle, Global eProcure, Zycus, Rosslyn Analytics, Ivalua, and Emptoris.

In December 2008, Ariba announced that the U.S. District Court for the Eastern District of Texas had issued an injunction against Emptoris, which prohibits the company from infringing on two of Ariba's patents related to overtime and bid ceilings in reverse auctions. On 16 December 2008, the court ordered Emptoris to pay an enhanced damages award of $1.4 million for willful infringement in connection with Emptoris' infringement of the two reverse auction-patents held by Ariba. This was in addition to the 29 October 2008 jury award of $5 million in damages to Ariba, bringing the total fine to approximately $6.4 million, a significant penalty for Emptoris which earned approximately $50 million in revenue for 2008. In an Emptoris press release, that company noted that it had released a new software "patch" that eliminates any infringement. The U.S. District Court, in February 2009, issued an order noting that the "patch" is colorably different, effectively concluding the case.

In November, 2010, Ariba announced that it would acquire Quadrem, a privately-held provider of one of the world's largest supply networks and on-demand supply management solutions. The acquisition closed in January 2011.

In October, 2011 Ariba announced the acquisition of b-process a privately-held French company and European leader in electronic invoicing service provider for approximately €35 million in cash.

2. SAP AG's acquisition

On May, 23rd, an Ariba Employee All Hands Call happened. We were told that SAP had bought Ariba for $4.3 billion. SAP now has a cloud computing partner in the competition against Oracle, its largest competitor.

It does not affect us and our project, but the prospective of being hired by Ariba at the end of the internship still exists. Here is an interesting article from Bloomberg about the acquisition:

"SAP AG (SAP), largest maker of enterprise- applications software, agreed to buy Ariba Inc. (ARBA) for $4.3 billion in the German company's second multi-billion purchase in cloud computing to take on Oracle Corp. (ORCL)

SAP will pay $45 a share, or 20 percent more than Ariba's May 21 closing price, Walldorf, Germany-based SAP said yesterday. The transaction, subject to approval by Ariba shareholders and regulators, will probably be completed by the end of August, SAP Chief Financial Officer Werner Brandt said on a conference call.

Ariba is the leader in cloud-based collaborative commerce applications, counting BHP Billiton Ltd. (BHP) and Deutsche Bank AG among customers it connects to more than 730,000 suppliers. As competition in on-demand software intensifies, SAP has increased its pace of acquisitions, last buying SuccessFactors Inc. in December. With businesses increasingly choosing to store and process data on the Web, SAP is shifting many of its staple applications to the Internet.

"We don't have the DNA in the cloud," SAP co-Chief Executive Officer Bill McDermott said in an interview. "We're probably the most strategic cloud player in the enterprise software industry."

SAP fell 0.9 percent to 47.38 euros at the close of trading in Frankfurt, paring the stock's increase this year to 16 percent. Ariba slipped 0.1 percent to $44.82 at 12:01 p.m. in New York, after jumping 19 percent yesterday.

Earnings Growth

In more than 60 enterprise software takeovers since 2002, the median multiple buyers paid was about 16 times earnings before interest, taxes, depreciation and amortization, according to data compiled by Bloomberg. That compares with the 106 times trailing 12-month Ebitda SAP has offered for Ariba, the data show.

Sunnyvale, California-based Ariba's Ebitda is projected to more than quadruple to $125 million in the 12 months through September, according to analyst estimates compiled by Bloomberg.

SAP's acquisition of Ariba would be the largest enterprise software deal since Hewlett-Packard Co. (HPQ) bought Autonomy Corp. for more than $10 billion last year, according to data compiled by Bloomberg. There have been almost 1,200 of those transactions globally over the past decade, with a value topping $80 billion.

Counterbid?

"There's potential for other bidders to emerge," said Richard Williams, an analyst at Cross Research in Livingston, New Jersey, who has a hold rating on SAP. "There's a history of bidding wars between SAP and Oracle and this is exactly the kind of strategic company that would spark something like that."

Deborah Hellinger, a spokeswoman for Redwood City, California-based Oracle, declined to comment. Oracle separately said today it agreed to buy Vitrue, a provider of cloud-based social marketing applications. Financial details weren't disclosed.

Ariba's global trading network connects and automates more than $319 billion in commercial transactions, collaborations, and intelligence among companies, according to SAP's statement. SAP's tools mostly serve internal business processes, and Ariba would give the German company a way to automate business transactions with outside companies and make it a stronger competitor in cloud-based business networks, Williams said.

The deal will add to SAP's profit, excluding some items, in 2013, Brandt said yesterday. Ariba CEO Robert Calderoni will join SAP's global management board.

SuccessFactors Takeover

SAP bought SuccessFactors in December 2011 for $3.4 billion, adding a maker of online personnel-management applications to help it enter the cloud-computing market. SAP is now offering its payroll, supply-chain management and financial software as cloud applications through the Internet.

The market for cloud software delivered via the Web will grow five times as quickly as sales of programs installed on clients' premises, according to researcher IDC. SAP plans to generate 2 billion euros ($2.5 billion) from such systems by 2015 to help it reach a sales target of more than 20 billion euros by that year.

While Oracle has snapped up rivals, SAP had traditionally shied away from large deals, with past CEO Leo Apotheker telling investors in 2009 that the company's home-grown software was superior to Oracle's. McDermott and co-CEO Jim Hagemann Snabe have changed that strategy as they race to snatch business from

Oracle and other companies such as Salesforce.com Inc. (CRM) and Workday Inc. in the cloud.

Hana Analytics

SAP bought Sybase Inc. for $5.8 billion in 2010 to bolster its database and mobile-computing offerings. It has also introduced a data-processing system called Hana, which competes with Oracle in database software.

This year, Oracle has purchased human-resources software maker Taleo Corp., a counterweight to SAP's acquisition of SuccessFactors, and customer-service cloud-software vendor RightNow Technologies Inc.

Ariba reported $444 million in revenue last year, an increase of 39 percent from a year earlier. The U.S. company's board has unanimously approved the deal, SAP said in the statement. SAP plans to fund the transaction with its own cash and a 2.4 billion-euro term loan facility, the company said.

At today's annual shareholder meeting in Mannheim, Germany, some SAP investors say the company may be paying too much. "We don't know whether Ariba can generate the profits that would justify the high price," Jella Benner-Heinacher, managing director of the German Association for the Protection of Wertpapierbesitz, said in an interview.

Ariba held its initial public offering in June 1999 and was one of darlings of the Internet bubble, seeing its stock more than quadruple in its first two months of trading. The shares plummeted about 90 percent in 2001.

JPMorgan Chase & Co. and Deutsche Bank AG advised SAP on the deal, while Morgan Stanley provided financial counsel to Ariba. "

II- Second part: project and internship

A- Context

This internship as a software engineering is a part of the completion of my master's degree in Computer Science in Aix-Marseille University. This internship takes place in Sunnyvale, California, United States of America. Sunnyvale is located in the middle of the Silicon Valley, the southern part of the San Francisco Bay Area in Northern California.

The region is home to many of the world's largest technology corporations. The term originally referred to the region's large number of silicon chip innovators and manufacturers, but eventually came to refer to all the high-tech businesses in the area; it is now generally used as a metonym for the American high-tech sector.

Almost 200 people work in Ariba's office in Sunnyvale, which is 16 years old; Last summer, Ariba's headquarters moved to its current position from a few blocks away. My internship was almost six months long from February 23rd to August 10th, 2012.

At the beginning, the project did not have any name. We figured that Arecibo, which is an observatory near the city of Arecibo in Puerto Rico, was an appropriate name for this project.

B- Team

Arecibo's team is composed of:

- Jacques Boutin: Senior Director of Engineering Program Office.
- Dinesh Shahane: Principal Staff Engineer, Architect, the project leader.
- Kingsley Ngan: Principal Software Engineer.
- Arnaud Hebert: Engineering Intern working on the front-end.
- Olivier Francome: Engineering Intern working on the back-end.
- Elmehdi Assir: Engineering Intern working on database aspect.
- Slade Capasso: Engineering Intern working on data analysis.
- Pallavi Otawkar: Senior Engineering Program Manager.

When we have meetings, Dinesh and Kingsley give us new instructions/guidance as we are working on tasks. When we recognize any issue or problem, they give us solutions to solve it. They already know where we are going; they keep us in the right direction.

But we have much interest for everyone's tasks, so we are working most of the time together. For example, our Group Tests, which I will talk about later.

What makes this project different is that the four of us did not start and finish our internships at the same time. I arrived first, in late February, and I left first in August. Olivier arrived in March, and left in late August. Elmehdi came in the beginning of April, and leaves at the end of September and Slade's 3-month internship occurred from June to August.

Pallavi joined us in mid-July. She is in charge of insuring the completion of the documentation, security, integration/deployment, etc… of our project.

C- Subject

3. Idea

Each day, thousands of transactions occur in Ariba's Network and it represents several million dollars being exchanged. Data is created by these transactions and it is a very important data source. But, in Ariba products, there are no applications that use or show this data.

The idea of creating an application that uses this data to show figures, statistics or maps based on real-time or historical data is born. We would like to show on a real-time map where transactions occur. For example, where the biggest areas are in term of amount or volume. Also, we would like to see historical data using cool visualizations. Everyone is using simple charts or graphs to show data, we would like to create innovative visualizations.

4. Description

The project involves real-time data from Ariba's cloud platform, aggregating data, and creating a real-time graphical representation of various network activities.

Examples include business transactions per second per location and between locations, spend in the Network, customer growth, business transactions between global regions, etc.

This is a team intern project with direct interaction with senior Ariba engineering leaders.

Technologies: Java, Tomcat, Apache, Hadoop, AribaWeb, HTML 5, Oracle.

D- Objectives

The main purpose of this internship is the building of an application from conception to production. The primary objectives are:

- Successfully compile, build, install and run the application.
- Design new functionality based on requirements provided by solution manager.
- Engage in collaborative discussions with other engineers.
- Implement new functionality as part of Ariba product according to Ariba coding guidelines.
- Write a unit test plan. Implement and run the unit tests.
- Troubleshoot and resolve defects found during the implementation and testing.
- Frequently present the progress of the feature through demonstration to the team.
- Develop understanding of development environment, the components and code base I will work on.
- Demonstrate good understanding of the tools.
- Complete the implementation of one or more features that passes quality criteria.
- Demonstrate good understanding of the development life cycle.
- Develop software implementation skills.
- Develop software design skills, testing skills and troubleshooting skills.
- Presentation skills.

E- Documents

1. Wiki

The team uses a Wiki to share documents, files and links related to the project. Here is the hierarchy of our Wiki:

- Planning
- Reviews
- Scrum Sheets
- Technologies
    - Technologies we use
    - Tested Technologies
- Tests
- Visualizations Examples Links

We had pages about technologies we are using. We are also working pages about technologies that we are not using but we tested: we explain why we didn't choose this technology and choose another one.

2. Blog

The team also writes a blog about our experience at Ariba, our experience in this internship and in this project. It is a way for Ariba's leaders to improve their experience of intern hiring.

We wrote an article about Arecibo's team, another dealing with our school and our internship program and another about the Arecibo project. We also want to help future foreign interns by writing the first things to do when arriving in the United States of America.

3. Repository

The Wiki was abandoned to the benefit of a repository in Ariba's server using Perforce for versioning. Our repository is organized in the following hierarchy:

- //ariba/devweb/ASM_Suite/versions/Visualizer/1.0/
  - ../DesignSpecs: Here is the Design Specifications Document: DSD.
  - …/Diagrams: Here are the diagrams of the application (UI, PO, and Architecture)
  - …/FRDs: Here is the Functional Requirements Document: FRD.
  - …/Miscellaneous: Here are the other documents about the application and the project.
  - …/Planning: Empty folder because no planning was set.
  - …/Reviews: Empty folder because no review was done yet.
  - …/Screenshots: Here are the screenshots of the application.
  - .../ScrumSheets: Here is the last version of ScrumSheet (everybody tasks)

- //ariba/cloudsvc/Arecibo

Here is the last version of the whole application code. In the "cloudsvc" folder are the other Ariba project codes.

We had to ask for permission to submit our work in the last folder and we didn't have to for the documentation folder ("devweb").

F- Project resolution steps

1. Needs analysis

Since it is an intern data visualization project, we didn't really have a needs analysis. At the beginning of the project, before starting anything, we spoke with Dinesh and Jacques who originally came up with the project. They were like clients, and we, the managers.

The main idea was, originally, to show "cool," real-time, visualizations, using JavaScript, HTML5 and CSS3, and "cool" historical visualizations using HBase. We already know that they wanted us to use HBase, but we need to find the best JavaScript libraries to do so.

In Ariba's project progress, the first step is supposed to be the writing of FRD but Dinesh didn't write it before late August. In this context, writing the Design Specifications Document was difficult because we were supposed to describe our architecture choices and our technology choices. We just spoke with Dinesh at the beginning of the project about technologies we were going to try, then, choose and validate.

2. Schedule planning

We are using Scrum method. Each of us have tasks to do, and we have a fifteen minute Scrum meeting twice a week with Dinesh, Kingsley and the four of us. Kingsley was leading the Scrum meetings while Dinesh was in India. We also define new tasks and talk about our work in progress.

We didn't have a real project planning. We worked as the project goes. We started the standard production, development process of a new Ariba product for the Arecibo project in early July.

3. Application development

The application development was continuous during this internship. I started by working on several prototypes of visualizations. When everybody on the project team had arrived, in early April, we started to build the application and use a versioning server.

The application development was stopped several times when we switched servers. We couldn't create the build anymore because somebody was modifying the architecture or the implementation of one or several classes.

4. Tests

a) Unit tests

For my work on front-end of the project, I had to use Web technologies such as HTML and JavaScript. When I create a new visualization, first, I tested it with static data (a JSON table). Then, I integrated this visualization to the application and tested it with the mock data in the CSV files. We created a Java class that generates the CSV files randomly. We tested the visualization with several series of CSV files.

b) Group tests

Our GTs was created when somebody makes a big improvement in the application and wanted everybody to install the new software or test the new feature. We wrote a GT file, a text file that contained instructions that were given in order for all team members to test the new feature. He had to report a bug, if any.

Here are quick descriptions of GTs that were created:

- The first GT main purpose was to install Apache Ant, Jetty Server. It was also a way to check if everybody has the JDK correctly installed and the environment variables set. Then, the user had to type an ANT command that created the build, launched the server and opened a web browser. The user needed to test the four first visualizations and report a bug, if any.
- The second GT goal was to test the HBase historical visualizations. HBase was installed on an Ariba Server called lucene01. A request is made by the client to lucene01 and HBase to send back data we need.
- The third GT gave instructions to install Virgo with Jetty, then instructions to create the application Bundle, deploy and run it.

5. Important meetings

We had several important meetings during this internship:

- The first meeting occurred May 4th, with Bhaskar Himatsingka (Chief Technology Officer) and Sanish Mondkar (Chief Product Officer) and the whole team (Jacques, Dinesh, Kingsley, Oliver, Slade, Elmehdi and I)
  We introduced ourselves to them and made a presentation composed of a slide show and a demonstration of our application. It was the first time the CTO and CPO saw the application. They were very proud of our work, and we discussed data privacy issues and improvements that can be made.

- Then, we had a meeting with Rick Collison on May 21st. The purpose of this meeting was to talk about data. We planned to get data from AN but Rick and his team have different data. He explained to us what kind of data they have and we told them what data we need and they figured if it was possible to get data from them.

- A meeting about Arches deployment occurred May 25th. Clayton Wilkinson, the engineer in charge of Arches, a similar Ariba project, which has almost the same implementation, demonstrated how they deployed Arches. We recorded this conference on video to watch it again because it is hard work and very complicated. We had to talk to RC team, OPS team and few others…

- The last meeting occurred in early August before I left. Dinesh (back from India), Kingsley, Pallavi, Bhaskar and Sanish attended. All of the architect engineers from Ariba's office in Sunnyvale were also at this demonstration and were shown the application. We talked about our progress and the current production step. They were very proud of our work and could not wait for when it is deployed.

6. Integration - Deployment

Our application integration and deployment tasks as an Ariba product were long and difficult. Elmehdi took care of this work.

The main repository of Ariba's application code is *//ariba/cloudsvc*. The dependencies main repository is *//ariba/3rdParty*. First step is to install the dependencies of our application: JSON libraries (we changed to the libraries that were already installed), Quartz libraries, Servlet libraries, HBase libraries, WebSocket libraries, JavaScript libraries and the application server.

Elmehdi followed the Arches integration and deployment plan because this other Ariba project is using Jetty Tomcat and its architecture looks like Arecibo. Because of the issues I will explain later about the relatively new WebSocket technology, we could not use Jetty Tomcat until the release came out in later September with WebSocket support. We asked for Virgo Jetty to be deployed, but the process was different from the Virgo Tomcat installation. The scripts for the Tomcat installation needed to be updated for a Jetty installation, the idea was abandoned and the application is still not deployed yet.

Installing a new 3rd Party such as new software is a huge process that needs to get approval from the RC team and the OPS team. We, as interns, asked for it and we don't know what's behind that point, because we are just developers.

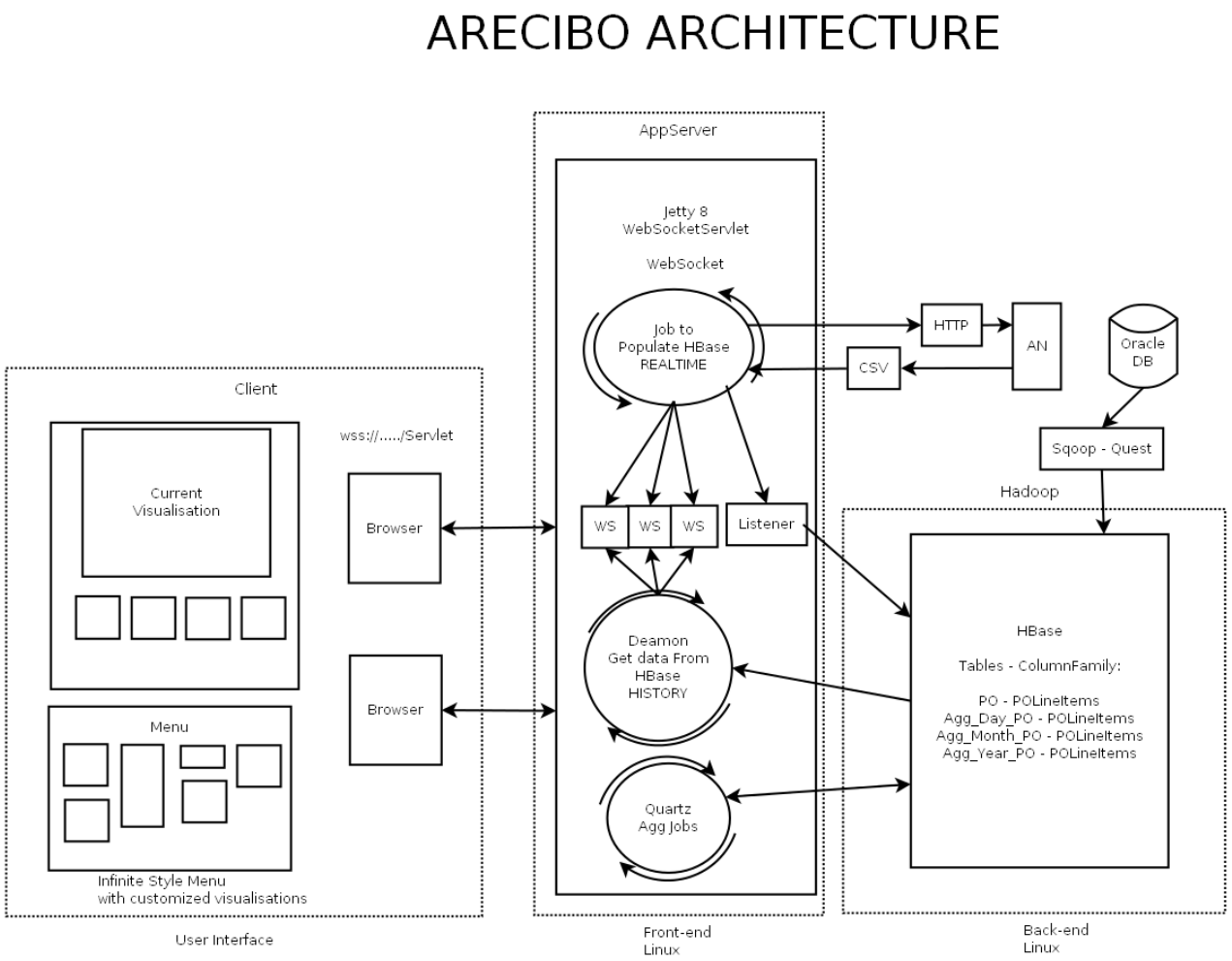
Another problem was that Ariba's servers can read JAR files, but not WAR files.

7. Production

Our application will not yet be available for Ariba's clients. First, Dinesh and Jacques plan to deploy the application on several television screens in Ariba's office in Sunnyvale and after security and privacy reviews, make it available for clients.

G- Application presentation

1. General schema



As previously mentioned, Olivier and Elmehdi were working more on the back-end side, Slade on the data analysis and I was working on the front side: the user interface. A description of the server, HBase and the User Interface will follow.

2. Server

At the beginning of my internship, I researched Node.js. Node.js uses an event-based server execution procedure rather than the multithreaded execution in PHP, for example. Node.js could still make use of its processing power when the server is waiting for any other operation. This makes Node.js scalable to millions of concurrent connections. Node.js works on a v8 environment – it is a virtual machine or a JavaScript engine that runs the JavaScript code, so for hosting you can't use the ordinary web hosts. You will need the ones that have the v8 environment.

But we wanted to utilize the server to do more than just that. For example, we wanted the server to be able to launch Quartz Scheduler Jobs (jobs whose tasks are defined as standard Java components that may execute virtually anything you may program them to do) and be able to set daemons to retrieve historical data from HBase.

So, we decided quickly to use another server: Jetty Server. Jetty provides an HTTP server, HTTP client, and javax.servlet container. These components are open source and available for commercial use and distribution. Jetty is used in a wide variety of projects and products. Jetty can be embedded in devices, tools, frameworks, application servers, and clusters. See the Jetty Powered page for more uses of Jetty. The core Jetty project is hosted by the Eclipse Foundation.

Why Jetty and not Tomcat? When we first needed a server, Jetty was the only one that provides a Web Socket API and our application communications between server and client are implemented with a Web Socket Servlet.

A few months after the beginning of our project, the new Tomcat release started supporting Web Socket Servlet. So, I modified the build in order to be able to launch the application either on Jetty or on Tomcat. Jetty and Tomcat APIs were quite similar so I only had to change few methods names and code.

Then, we wanted to make another big improvement in our application: the application will be a component itself and each visualization must also be a component on its own. With this structure, the application will be scalable so we can add or remove visualization easily, and if some other developer wants to create a new visualization, he will be able to add his bundle to the application.

Also, the application requires many libraries such as JSON, Quartz, Jetty, Servlet and Hadoop.

Virgo Server is the perfect tool for us. It comes with Jetty (or Tomcat), Spring and OSGi. The Virgo Web Server from EclipseRT is a completely module-based Java application server that is designed to run enterprise Java applications and Spring-powered applications with a high degree of flexibility and reliability. It offers a simple

yet comprehensive platform to develop, deploy, and service enterprise Java applications.

3. Communication client/server

Our application needed technology providing bi-directional communications. The server sends each second, a packet of real-time data, to the clients, and the client can ask the server for historical data. Choosing WebSocket was obvious.

"WebSocket is a web technology providing for bi-directional, full-duplex communications channels over a single TCP connection. The WebSocket API is being standardized by the W3C, and the WebSocket protocol has been standardized by the IETF as RFC 6455.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket protocol makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time games. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. A similar effect has been achieved in non-standardized ways using stop-gap technologies such as Comet."

WebSocket is a new technology, so we encountered problems with its support on servers. At the beginning, we used Jetty that provides WebSocket support and Tomcat does not. Then, Tomcat's new release supported WebSocket so we switched to Tomcat because most of Ariba's products are based on Tomcat. This would make the integration deployment easier. Then, we changed to Virgo Jetty because we needed Spring and OSGi support, but the integration deployment encountered many issues. We wanted to switch back to Jetty Tomcat, but the pre-packaged Tomcat that supports WebSocket will not be released until late September.

Here is sample of code on the server side (Java file):

```
public class VisualizationServlet extends WebSocketServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }

    public WebSocket doWebSocketConnect(HttpServletRequest request, String response) {
        StreamingSocket ss = new StreamingSocket();
        return ss;
    }

    public class StreamingSocket implements WebSocket.OnTextMessage {
        private Connection connection;

        public void onClose(int arg0, String arg1) {
               // CODE
        }

        public void onOpen(Connection connection) {
               // CODE
        }

        public void onMessage(final String data) {
               // CODE
        }

        public void onData(JSONObject data) {
               // CODE
        }
    }
}
```

Here is sample of code on the client side (JavaScript file):

```
var wsUri = "ws://localhost:8080/arecibo/Main";
var websocket = null;

function streamingWebSocket(){
	websocket = new WebSocket(wsUri);
	websocket.onopen = function() { onOpen(); };
	websocket.onclose = function() { onClose(); };
	websocket.onmessage = function(evt) { onMessage(evt); };
	websocket.onerror = function(evt) { onError(evt); };
}

function onOpen(){
	// CODE
}

function onClose(){
	// CODE
}

function onMessage(evt){
	// CODE
}

function onError(evt){
	// CODE
}

function doSend(msg){
	// CODE
}
```

4. HBase from Hadoop

a) The Data Base

This is a data based project. Data is pulled from AN each second. We need this data for real-time visualization and we need to save this data for historical visualizations.

Every day, thousands of transactions are made, and it represents thousands of rows of data.

Using data base HBase that comes with Hadoop was obvious. We need random, real-time read/write access to our Big Data. Hadoop project's goal is the hosting of very large tables -- billions of rows by millions of columns -- atop clusters of commodity hardware. HBase is an open-source, distributed, versioned, column-oriented store modeled after Google's Bigtable: A Distributed Storage System for Structured Data by Chang et al.

For the development, we used a standalone HBase, but when application is in production with several clusters, it is more efficient.

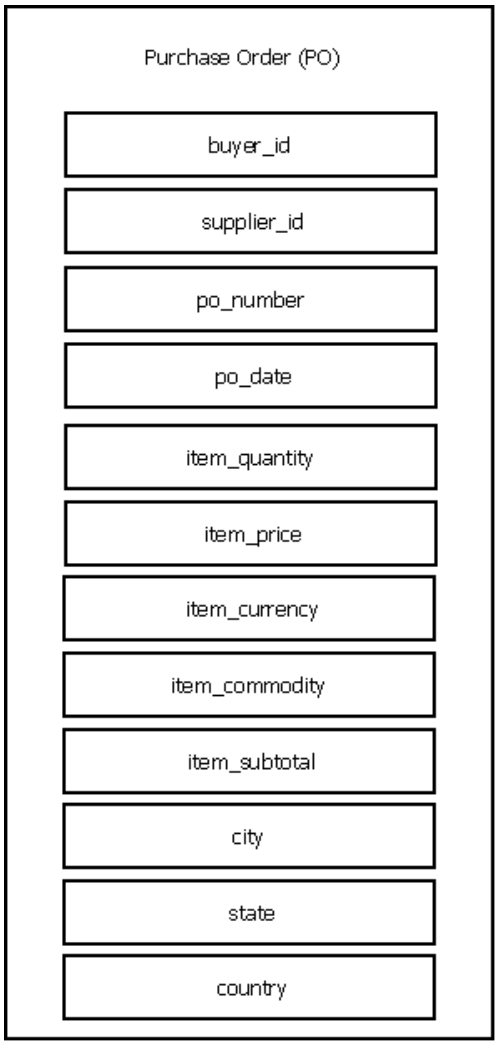
b) Data

At the beginning, we didn't know much about data that we were going to get from AN. We were told that we will be working on data called POs: (purchase orders). These are transactions made by buyer and seller, both Ariba's clients. We assumed that POs were made of amount, volume, supplier location and buyer location. It was enough to start building interesting visualizations.

We didn't get real data from AN, so we used a different channel to get data; we created CSV files with mock, but consistent, data.

The structure of the application is done in order to get real-time data from different sources. We have a CSV Data Source and HTTP Data Source.

Here is a preview of a PO:



- buyer_id: buyer's id in AN
- supplier_id: supplier's id in AN
- po (PO_NUMBER) : PO's ID in AN
- po_date: PO's date
- item_quantity: quantity of product sold/bought
- item_price: PO's price
- item_currency: PO's currency
- item_commodity: type of product sold/bought
- item_subtotal: price of everything purchased before taxes
- city: city of the buyer
- state: state of the buyer
- country: country of the buyer

The HBase structure is very simple: only one table: "PO". The row key is generated with the current timestamp at the first insertion of the series. It matches the pattern yyyymmddhhmmssSSS where SSS are the milliseconds. It is incremented for each insertion. There's only one column family as well: "POLineItems", which regroup every characteristic of the PO.

There are three other tables that will contain the aggregated data based on days, months or years aggregation. These qualifiers are created at each insertion:

- Agg_Day_PO
  - row_id (date)
  - volume
  - amount

- Agg_Month_PO
  - row_id (date)
  - volume
  - amount

- Agg_Year_PO
  - row_id (date)
  - volume
  - amount

The HBase structure can be created by the following commands:

- create table 'PO', 'POLineItems'
- create table 'Agg_Day_PO', 'POLineItems'
- create table 'Agg_Month_PO', 'POLineItems'
- create table 'Agg_Year_PO', 'POLineItems'

It contains one line per POLineItems (One line by PO).

A data request is sent by an HTTP request in 10-minute intervals to AN. The system calls the URL with the proper time as a parameter:
https://parakeet.ariba.com:8450/ANBusinessProcessMonitor.aw/ad/monitorPoItem?beginTime=2012-04-10T01:53:03:-08:00

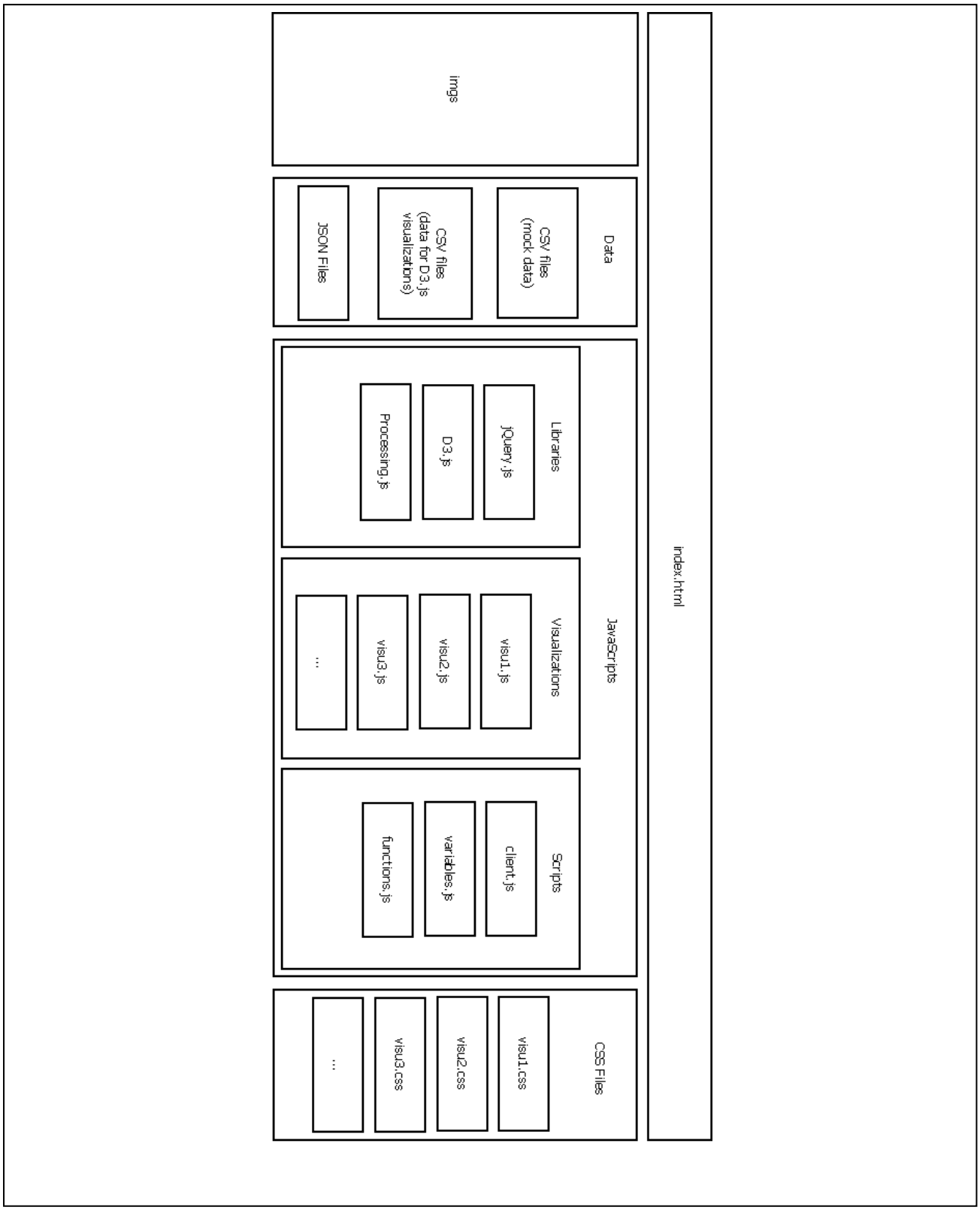
Response is a CSV file; the daemon running in the server saves data into HBase.

The location data is translated into a geo-code (using a cache table or asking the Google Web Service Geocode) to send it to the listeners.

As the application cannot show current quarter information, delayed PO data will be requested from the URL (90 days).

5. User Interface

Here is a schema of the user interface architecture.



A menu allows the user to choose, among twelve visualizations, the one he wants to see. On the menu shows thumbs of all visualizations which are clickable. On a click, the menu disappears and the chosen visualization appears in full screen. On each visualization, there is a button "Go back to menu" in order to the client to go back to the menu. Several Ariba design (Logo and header) were added to our application menu, it's a design reference to other Ariba products.

All data is sent from the server to the client. We implemented JavaScript functions for each visualization so that only data needed is used for the visualization.

c) Visualizations based on Real-time data

Visualizations are updated every second; the client receives data from the server.

- Google Maps Markers:

This visualization is simple. First, a blue marker was placed where buyer is located and a red marker is placed where supplier is located for each PO. A Google Maps button allows the client to visualize both, only buyers or only suppliers. A key at the bottom of the page shows what marker represents.

But we were told that this visualization was not respecting buyer and supplier privacy. I modified this visualization in order to place same color markers on buyer and supplier location and just tell the client that transaction but you do not know what kind happened there. Also, we were told five months after the prototype that we could not get supplier location.

- Google Maps Circles:

Another simple visualization, a circle is placed where supplier is located for each PO. The size of the circle depends on the transaction volume or the transaction amount. A Google Maps button allows client to switch from volume to amount. We quickly found a special case, when several suppliers are located in the same area. Then, we implemented aggregated transaction tables in order to make the circle bigger instead of having several little circles in the same area. A key at the bottom of the page shows how much represents a small circle and how much represents a big circle.

But we were told that this key was not respecting buyer and supplier privacy. I modified this key in order to say that big circles represent a big transaction and a small circle a smaller transaction, so, we avoid showing figures.

- Google Maps Line:

Similar to the previous two simple visualizations, a marker is placed where buyer is located and another marker is placed where supplier is located. Then, a line is drawn between buyer and supplier for each PO.

Previous lines disappear when new data arrives from server and new lines appear. Dinesh wanted the lines staying longer, so, the lines now stay a few seconds and I added a Google Maps button that allows client to reset all lines.

- Google Maps Curves:

It is almost the same visualization as the previous. It is not a line that makes the link between buyer and supplier for each PO, but a curve. The curve is created using Google Maps API. Google Maps curve is the shorter way from one point to another point on earth. For example, the shorter way to go from Paris to San Francisco is to go by Greenland. Dinesh wanted the curves to stay longer as well. I made the curves stay a few seconds and I added a Google Maps button that allows client to reset all curves.

- D3.js curves:

This visualization uses Google Maps and D3.js. It represents transactions by on motion curves on a map. Curves are still the links between buyers and suppliers. The previous visualization was based on Google Map API curves but curves where static. On this visualization, curves are drawn on top of the Google Map on an Overlay Map element. Fortunately, Google Map API is very simple; there are functions to convert latitude/longitude position to pixel position on the overlay map. This visualization was very hard to build, and the number of transaction is limited because the web browser does not deal with many transactions.

- Processing.js curves:

This visualization is almost the same visualization than the previous but it uses Processing.js. Processing curves cannot be drawn on top of a Google Maps Overlay Map object such as D3.js. For this visualization, a static map is used. Curves are fully drawn each second and a little point goes point by point from the buyer to the supplier (from 0 to 100%).

- Slot machine using CSS margins attribute:

This visualization uses HTML5 and CSS3. This is a slot machine. We show a mock amount of money that client saves from the time he load that page (you saved 10,000$ since 65 seconds on this page). It uses div element and margin-top attribute to make numbers go up such as a slot machine. Every second, a function adds to the value of money saved by the company, the counter is incremented and the slot machine shows the increasing value.

- D3.js Stream Graph:

This visualization is based on D3.js. We want to see a real-time graph of currency of transactions that updates each second. This graph has real-time data entering from the right and has old data leaving from the left and I added text labels that tell the user which stream represents which currency. I also worked on adding a built-in axes and time scales but for privacy issues, the idea was rejected.

- D3.js Ariba Top Commodity Wheel:

This visualization is also based on D3.js. Top commodities hierarchy organized in a sunburst tree. A commodity flashes whenever a Sourcing project is created with it. A click on a commodity makes a zoom in, and a click on the center circle makes a zoom out.

- D3.js Ariba Network Connections:

Such as the two previous visualizations, this visualization is based on D3.js. AN connections by country in a chord diagram.
A mouse over a country on the edge is showing connections of that country.

d) Visualizations based on historical data

- Google Chart Tools:

This visualization uses Google Chart Tools. It makes charts with historical data. There are nine buttons that correspond to ranges: seven days, fourteen days, one month, three months, six months, one year, two years, five years and max. When you click on a button, the client sends a request to the server and the server sends back aggregated tables (day, month and year) that corresponds to ranges: data within fourteen days, for example. Then, the chart is made and shown with data we received. I also made a function that rebuilds the aggregated tables with all days because some days, no transactions occurred. Above the chart, I added statistics: amount/volume/number of transaction for this range in percentage of the month, and in percentage of the year. There are two additional buttons for big ranges: for example, five years of transactions can be displayed per day, per month or per year. These buttons are not enabled when range is seven days, fourteen days and one month.

- Google Chart Tools with on-demand range:

It is almost the same visualization as the previous, but in this one, we get max data that we have from HBase and all aggregated PO's data. By default, we have a maximum range of time, but with Google Map Chart Range Filter, you can reduce the range of time that you want to see with a simple drag and drop.

6. Application goals

The goals of our project are defined such as:

- Real-Time Commerce / Ariba index (Real-time)

Real-time goal was achieved at a certain point. We are getting data in real-time, but for security concerns, it cannot be "true" real-time data but only delayed data.

- Comparative analysis / Trends (Historical)

This goal is achieved. We can get data from HBase, but historical visualizations need to be reviewed for data security concerns.

- BHAG: Big Hairy Audacious Goal (predictive)

We did not work much on that because of the difficulties with gaining access to data sets needed to research and create predictive algorithms. They were supposed to come from an Ariba office outside the US but did not get uploaded before my internship ended.

7. Others useful tools

e) Perforce

We are using Perforce for versioning.

Perforce is a leader in enterprise version management. Perforce takes the conflict out of collaboration by making it easy for teams to find the right digital assets, work on them in parallel and store them securely. We can version everything with Perforce.

When we wanted to add some stuff to the repository of the project, we asked for access to submit our work. We were given Perforce access for several hours by Kingsley or Clayton. So, we had to ask them many times.

f) IntelliJ Idea

We are using Idea, it is a great IDE to work with, and very simple to use with the Perforce plug-in. Idea is a commercial Java IDE by JetBrains. It is often simply referred to as "IDEA" or "IntelliJ". Idea supports a lot of features and technologies such as Java, JavaScript, HTML and CSS, the technologies we are using. It is a good way to save time and avoid syntax errors.

g) Apache ANT

We are using ANT.

Apache ANT is a software tool for automating software build processes. It is similar to Make, but is implemented using the Java language, requires the Java platform, and is best suited to building Java projects.

Our build is very complicated and we could not do it manually every time we want to test something. ANT was the tool we needed. A build.xml configuration file was written which made the build of the application easy and fast. It is also a good way to save time and avoid build errors.

H- Application coding details

1. « Core » package

In this package are the classes related to the core of the application. The data sources classes will implement the interface "Data Source".

```
public interface DataSource {
    abstract void addListener (DataSourceListener listener);
    abstract void removeListener (DataSourceListener listener);
    abstract String accessData(int time);
    abstract List<DataSourceListener> getListeners();
}
```

As shown on the screenshot above, a Publish & Subscribe design pattern was implemented, where clients will subscribe to the data sources they need and become listeners. Then, the data source will push data to clients that have subscribed.

h) a) « Data Source » interface

This interface groups all methods that will be implemented in the data sources classes together.

- Add Listener: a method that adds the client as a listener to the application when the client connects.
- Remove Listener: a method that removes the listener to the application when the client disconnects.
- Remove All Listeners: a method that removes all the listeners from the application.
- Access Data: a method that accesses data from the source and returns a JSON Object.
- Get Listeners: a getter for list of listeners.
- Get Aggregated Data: a getter for data which returns a JSON Object.

i) b) « Data Source Listener » interface

This interface groups all methods that will be implemented in the data source listeners' classes together.

- On Data: a method which be called when data will be received.

j) c) « Data Source Service » class

The following is a description of this class:

- Register CSV Data Source: a method that creates the CSV Data Source instance.
- Register HBA Data Source: a method that creates the HBA Data Source instance.
- Unregister CSV Data Source: a method that destroys the CSV Data Source instance.
- Unregister HBA Data Source: a method that destroys the HBA Data Source instance.
- Get CSV Data Source: a getter that returns the CSV Data Source instance.
- Get HBA Data Source: a getter that returns the HBA Data Source instance.

A singleton design pattern was implemented for each of these two data sources in order to make sure that they will not be more than one data source of each type instantiated. A "Data Source Service" class was created and will return the current instance of the data source type requested.

```
public class DataSourceService {
    static DataSource DataSourceInstance;

    public static void registerDataSource(DataSource ds)
    {
        DataSourceInstance = ds;
    }

    public static void unregisterDataSource(DataSource ds)
    {
        DataSourceInstance = null;
    }

    public static DataSource getDataSource ()
    {
        return DataSourceInstance;
    }
}
```

2. « Server » package

In this package, are the classes related to the application's server.

k) « Compare Date » class

This class compares two dates together that returns 0 if the dates are the same, otherwise the difference.

l) « CSV Data Source » class

This class implements methods from Data Source class in "core" package for CSV data source.

m) « HTTP CSV Data Source » class

This class implements methods from Data Source class in "core" package for HTTP CSV data source.

n) « Day Aggregated Job » class

A Quartz Job that will be done each day. An aggregated day table will be updated by this job each time we have new transactions.

o) « Month Aggregated Job » class

A Quartz Job that will be done each month. An aggregated month table will be updated by this job each time we have new transactions.

p) « Year Aggregated Job » class

A Quartz Job that will be done each year. An aggregated year table will be updated by this job each time we have new transactions.

q) « Populate HBase » class

This class is designed to populate HBase when there are new transactions. It creates a row for each transaction and inserts the row in the HBase. The insertion of the Real-Time data in HBase is made by this class as a particular listener on the CSV data source. Its job is to listen to the real-time data (it only subscribes to the CSV data source) and to insert the data received into HBase. The insertion key is the sum of the timestamp of the first insertion and the number of insertions already done. It's also in charge of processing data to fill the different aggregated tables in HBase.

3. « Main » package

In this package, we have the main class.

r) « MANIFEST.MF » file

This manifest file is designed for the application bundle. When we build the application, this configuration file is needed for the jar file that is created.

s) « Main » class

The main class is where everything begins. This is where we initialize everything on the server side: Data Source Listeners, the Thread and the Quartz Jobs.

- Thread is getting data from CSV files (mock data) every second.
- Quartz Jobs are the aggregating jobs for tables.
- Data Source that will send data to Listeners when they register.

t) « Resource » folders

There are two resources folders for Jetty and Tomcat deployment. They both need libraries, a file web.xml for the Servlet mapping and an Arecibo.xml for the web application context mapping.

4. « Ui » package

In this package, are all the files needed for the user interface of the application.

u) « Visualization Servlet » class

There are two versions of this class. One for Tomcat one for Jetty because their API are different. When we build for launch on Tomcat server or for launch on Jetty server, he picks up the corresponding class. This class is the Web Socket Servlet: it implements the communicate methods from server to client:

- On Open: a method that opens the Web Socket connection between client and server.
- On Close: a method that closes the Web Socket connection between client and server.
- On Data: a method which is called when server receive data from client.
- Send Message: method that sends message from server to client.
- On Text Message: a method which is called when server receive message from client.

v) « Data » folder

In this folder, we have the CSV files that are read each second by CSV thread. The files contain the mock data. We are using this data source before we get real data from AN.

w) «Libraries » folder

We also have libraries that are needed for compilation (Tomcat libraries for Tomcat Visualization Servlet).

x) « Resource » folder

This folder contains all the web content:

- Html files.
- CSS files.
- JavaScript files.
- CSV and JSON files for D3.js visualizations.
- Images.

5. « Bin » folder

y) « Build Bundle.sh » file

This file can be called to build the application bundle (a run able jar file) that can be deployed on a server such as Jetty or Tomcat.

z) Group tests files

Here are the group tests files, txt files that contain instructions that were given for each group test in order for all team members to test the application. For example, group test number three gives instructions to install Virgo with Jetty, then instructions to create the application Bundle and then deploy and run it.

aa) Configuration files

Several files (sh or bat) are called automatically by the build to set up your environment variables. Also, others scripts files can be called to start the application on Tomcat, another one, to start the application on Jetty and then to stop these servers.

6. Build

Each package (core, server, ui and main) has its own build.xml file that creates a jar file using ANT. They are low-level ant build scripts.

The main high-level ant build script creates a war file of the application. The war will be deployed onto Jetty or Tomcat. Because of the dependencies, the application must build each jar file in the following order:

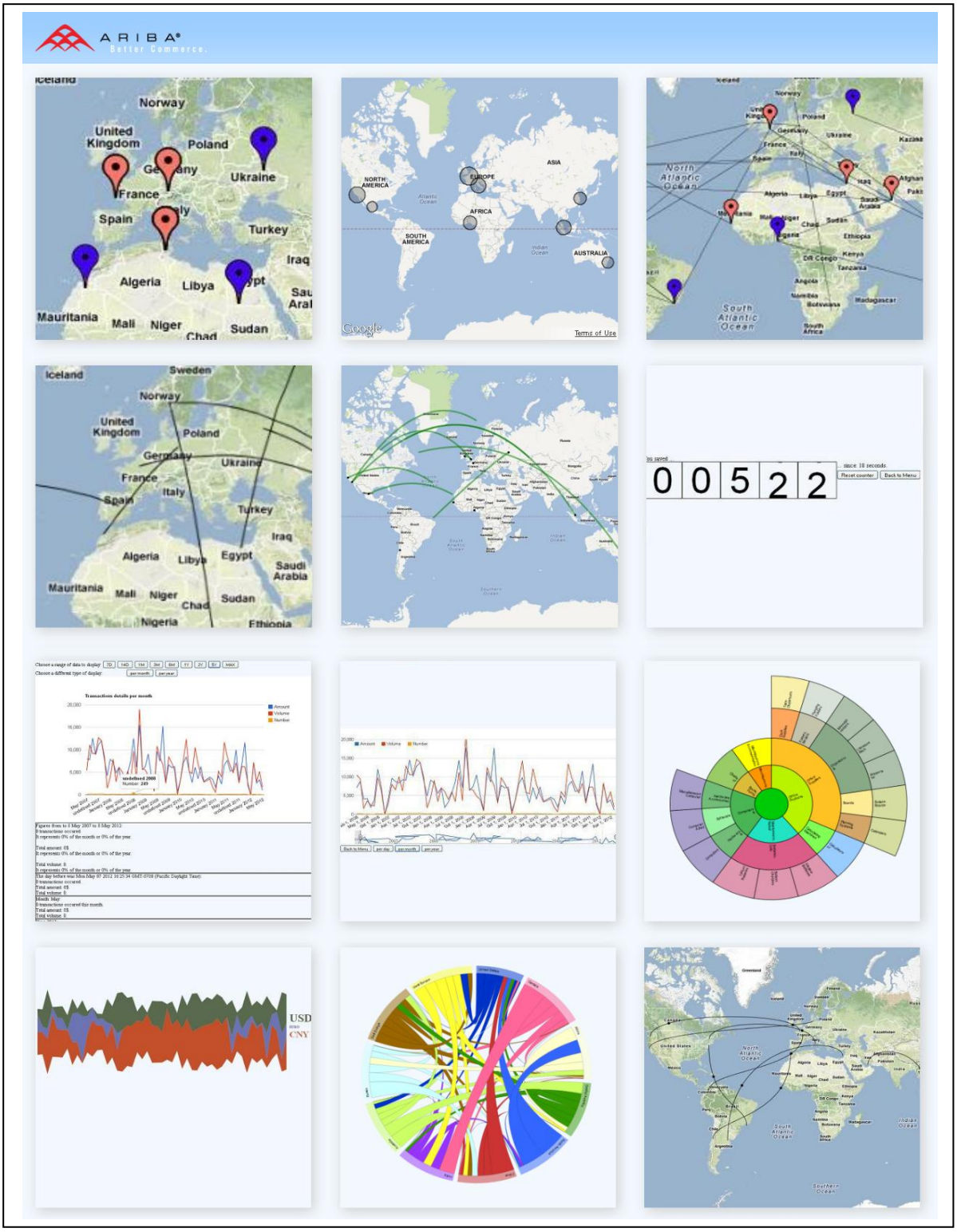
- core package
- server package
- ui package
- main package

The build creates the runable application jar and includes the resources files such as the whole UI: HTML files, JS files, CSS files, mock data files, and images.

I- Results

1. Application presentation

Here are several screenshots of the application. Here is the menu:



2. Example of visualization

See all visualizations screenshots in the Appendix.

3. Possible improvements

About the existing menu, it can be improved by replacing the static thumbnails of several visualizations by little previews of moving real-time visualizations. It was done for the Google Maps visualizations, but it was not done for the other visualizations.

Another visualization that we thought of but we did not have time to do is, for example, a graph showing a comparison between Ariba's activities and the stock exchange rate.

The idea of adding an interactive device came in the early beginning of the project. It is going to be a Leap device ("Kinect-like") and we already found the libraries we are going to use. But we did not have time to make it concrete. Unlike the Xbox Kinect and Nintendo Wii sensors bars that perch atop your screens, the Leap plugs into your USB port and loads gesture control software onto your device; after calibrating the system, the Leap tracks your every movement, recreating those movements on screen. A simple thumb drive, the Leap sensor is much smaller than the Kinect bar. It is also 200 times more accurate and able to follow your movements to the 1/100th of a millimeter, according to Leap Motion's website.

About the data, we had some meetings with other Ariba engineers working on other sources of data. If this project gets another data source, we might be able to build some other visualizations using different kind of data.

We also worked on historical data visualizations that cannot be displayed yet. We need to aggregate data and see if any security and privacy concerns arise from these visualizations before they can be enabled.

Another goal of the application will be the predictive aspect. If we work with historical data, we might be able to estimate the future data and show it in some way.

The application runs with bundles. Each package in the application is a bundle. The four bundles (core, server, ui and main) are considered the foundation of the application because without them, the application doesn't work. The other bundles represent visualizations (on one bundle per visualization) that can be enabled or disabled by the user. If some other developers want to create and add a new visualization, they just have to deploy the bundle on the server.

J- Conclusion

1. Working in Ariba

Working in Ariba, in America's Silicon Valley, was a very good experience for me. It was a chance to be part of a large, growing company in a prestigious region, the leading hub for high-tech innovation and development. Others engineers were very friendly and we often discussed with them about computer science related subjects and sometimes not. It was interesting to learn from them. Facilities are very comfortable, computers are quality, the cafeteria is a good place to meet and have lunch. We can go to the gym anytime we want, and everything is made such as you feel like home.

2. Internship

The project was very interesting; we built the application from scratch. We created everything. I was pride of our application and our work when we made demonstrations.

It is true that when you first heard about what Ariba does, you don't really understand what they do. We learned much about Ariba from data and also from discussions with others engineers. We also attended meetings such as "All hands" meeting where they talked about product organization, engineering organization and SAP's acquisition.

I learned much about project organization and project deployment with this project. I learned from others team members

Also, we went to several of the weekly "Engineering Brown Bag" sessions. Each week, a guest speaker talks about an engineering related subject. For example, there was a session about Node.js and another about jQuery.

3. School contribution

What I learned at school was very useful for this internship. I improved my knowledge in JavaScript, HTML5 and Java, but my skills for these technologies were a pretty good base for this internship.

Also, our skills in project management and project organization were needed for this internship. Each team organization is different from one enterprise to another enterprise and even sometimes from one project to another project. The team was composed of four interns, each of us had different skills and different work to do, but we had to communicate to integrate our work to the application. For example, my front-end work depends on the back-end data, Olivier's work.

4. Problems found

At the beginning of the internship, I just arrived in US. It took me several days to adapt to the English language. In Ariba, most of employees are from all around the world. Most of them are from India and China, so everyone speaks with an English accent. By the end of my internship, I was bilingual.

This project deals with Ariba data, POs from supplier to buyer. We only got this data from AN after a few months. We cannot do whatever we want with this data due to security issues that needed to be addressed, so we always worked with mock data, data that we invented ourselves. When my internship at Ariba ended, the application was still a prototype.

We worked several month assuming we could get supplier information such as location (latitude and longitude). But in fact, data that will come from AN will not provide everything we thought. So, most of our work on data was based on assumptions that were incorrect.

I worked on many visualizations using JavaScript libraries such as D3.js, Processing.js and Node.js. Sometimes, due to a lack of documentation, I had some issues that kept remaining unsolved.

Dinesh was away in India for two or three months for other Ariba projects he is working on, so we learned to work by ourselves in autonomy. Kingsley was leading the Scrum meetings and he helped us when we needed.

5. Conclusion

The amount of knowledge gained during my internship in Internet/Cloud Computing technologies, product development processes, design and architecture, coding skills and, testing and validation is vast.

Things could have been easier if there were no other projects to distract our project leaders. But my co-workers were very helpful, answering my questions and helping with assignments. We encountered many issues about data that were slowing the project development and also many issues about integration deployment that the project had to remain a prototype.

I am disappointed that I did not see the application running in production and people using it and have real feedback from users. I only saw the application as a prototype and the only people that saw it were the people attending demonstration meetings.

Also, I found in my internship in Ariba, all what I wanted and all my expectations were fulfilled.

I would recommend Ariba's Internship Program for the next year's students that can be interested in an internship in a big company in Silicon Valley, in United States America.

And I am very interested in working for Ariba again.

K- Appendix

1. Bibliography

Wikipedia http://en.wikipedia.org/wiki/Main_Page

Ariba Website http://www.ariba.com/

Jetty Server http://jetty.codehaus.org/jetty/

Tomcat Server http://tomcat.apache.org/

Virgo Server http://www.eclipse.org/virgo/

Spring http://www.springsource.org/spring-framework/

OSGi http://www.osgi.org/Main/HomePage

Hadoop http://hadoop.apache.org/

HBase http://hbase.apache.org/

Article about SAP acquisition

http://www.bloomberg.com/news/2012-05-22/sap-agrees-to-buy-ariba.html

Arecibo Observatory http://en.wikipedia.org/wiki/Arecibo_Observatory

Node.js Server http://www.hongkiat.com/blog/node-js-server-side-javascript/

Quartz Scheduler http://quartz-scheduler.org/

Perforce http://www.perforce.com/

IntelliJ Idea http://www.jetbrains.com/idea/

Apache ANT http://ant.apache.org/

WebSocket http://en.wikipedia.org/wiki/WebSocket

2. Vocabulary

BHAG: Big Hairy Audacious Goal is a strategic business statement which is created to focus an organization on a single medium-long term organization-wide goal which is audacious, likely to be externally questionable, but not internally regarded as impossible.

Software framework: In computer programming, this is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code, thus providing specific functionality. A Framework is a package of software libraries that abstracts computer programs by wrapping them in a well-defined and reusable application programming interface (API). However, frameworks contain key distinguishing features that separate them from normal libraries.

SAP AG: a German software company.

SAP ERP: software made by SAP is the world leader in ERP software in enterprise (99% of enterprises use SAP ERP)

ERP: Enterprise Resource Planning integrates internal and external management information across an entire organization, embracing finance/accounting, manufacturing, sales and service, CRM, etc. ERP systems automate this activity with an integrated software application.

VM: Virtual Machine.

IDE: Integrated Development Environment.

IT: Information Technology.

OSS: Open Source Software.

JDK: Java Development Kit.

JRE: Java Runtime Environment.

KPI: Key Performance Indicators.

JSON: JavaScript Object Notation.

JAR: Java ARchive.

WAR: Web Application Archive.

API: Application Programming Interface.

OSGi: Open Services Gateway initiative framework.

ANT (Apache): « Another Neat Tool ».

SaaS: software-as-a-service.

FRD: Functional Requirements Document.

DSD: Design Specifications Document.

GT: Group Test.

UI: User Interface.

WUI: Web User Interface.

PO: Purchase Order.

AF: Ariba Framework.

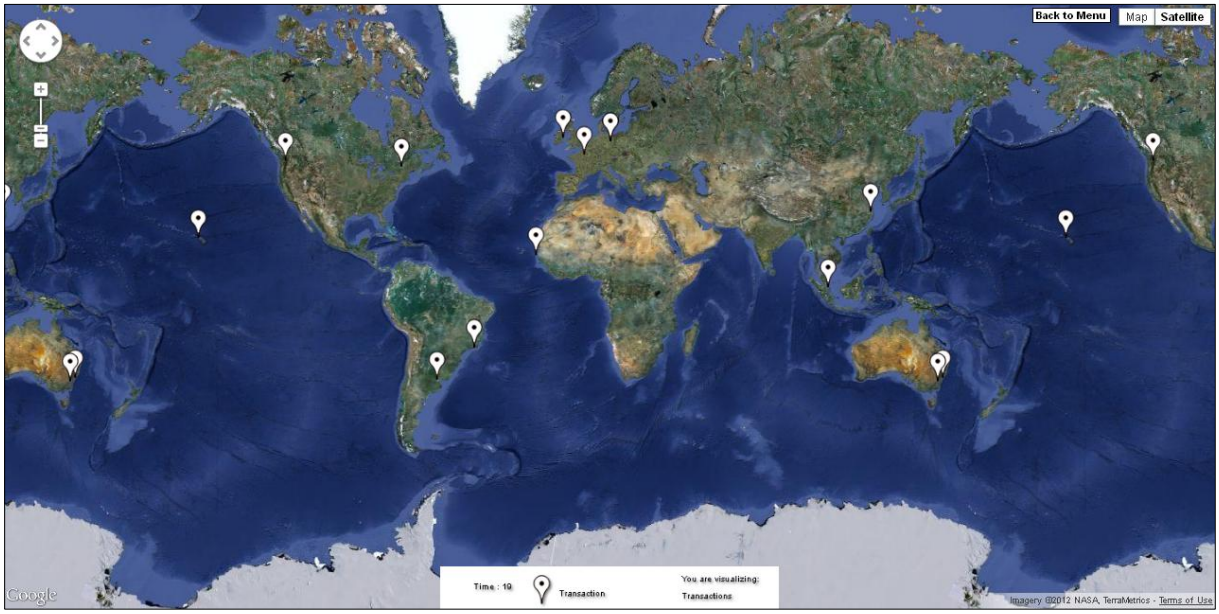
AN: Ariba Network.

RC: Release Control.

OPS: OPerationS.

3. Visualizations screenshots

Here is visualization 1:



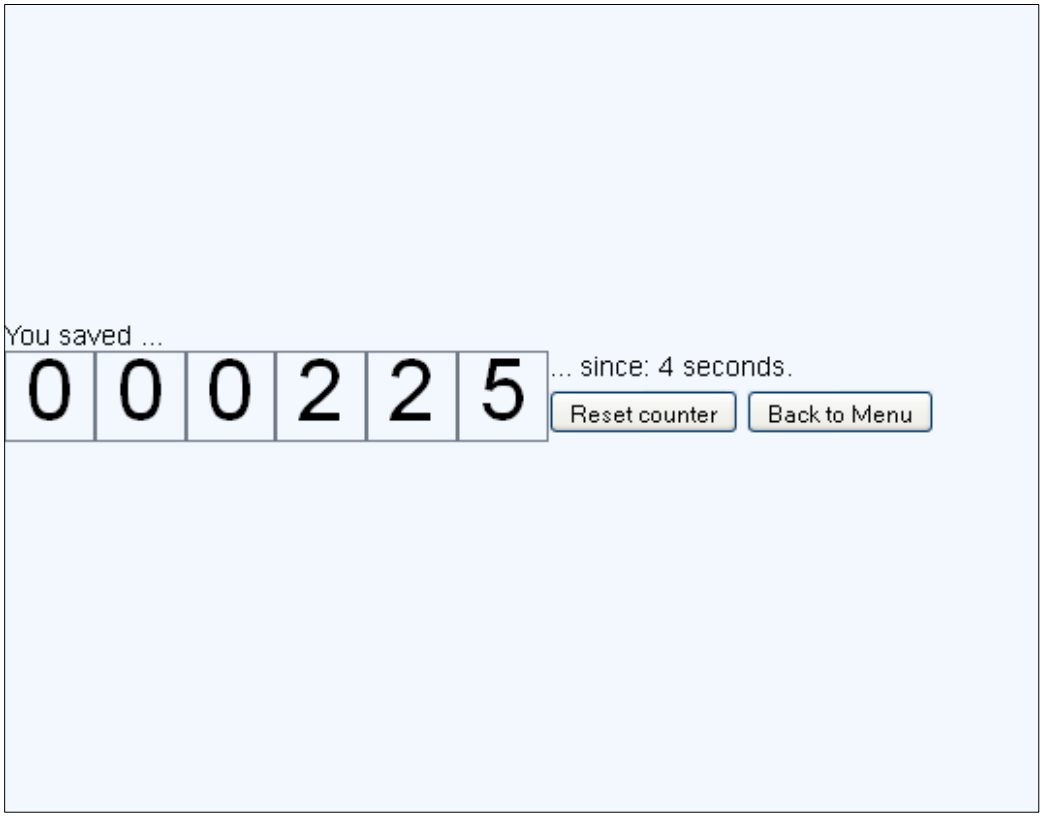
Here is visualization 2:

Here is visualization 3:

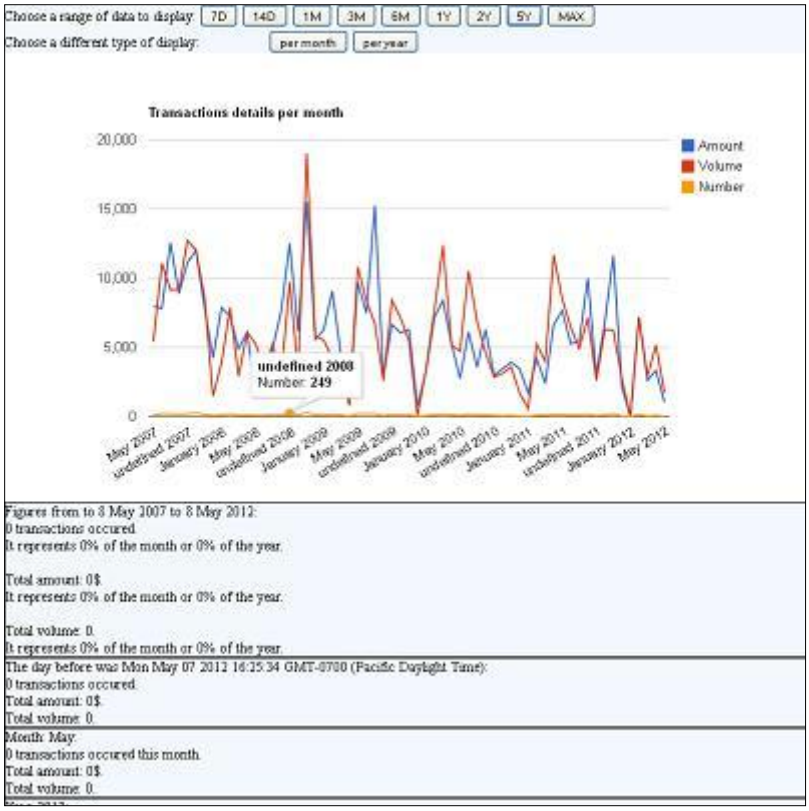
Here is visualization 4:

Here is visualization 5:

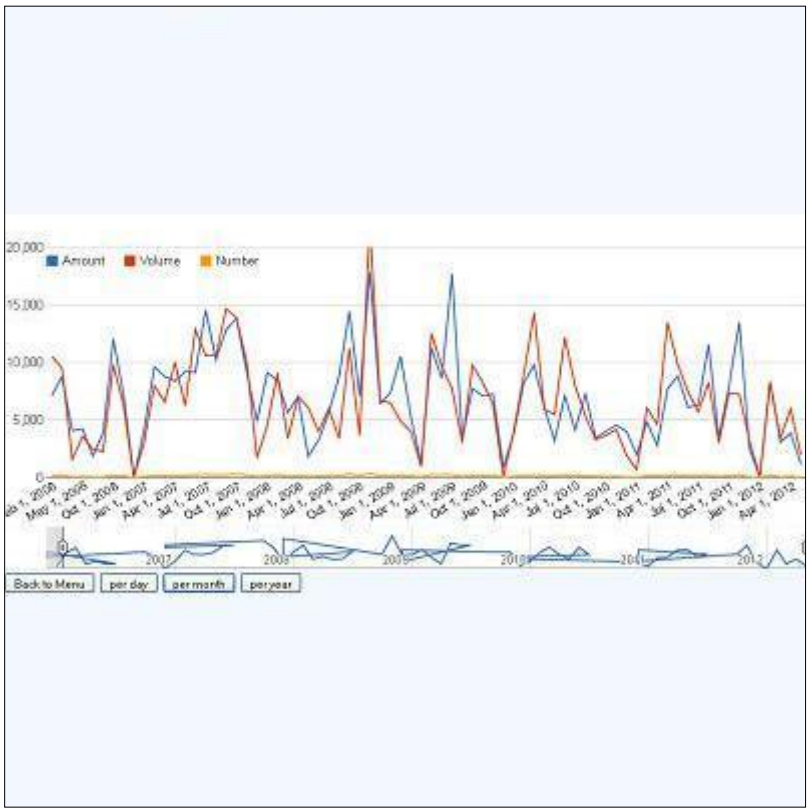
Here is visualization 6:



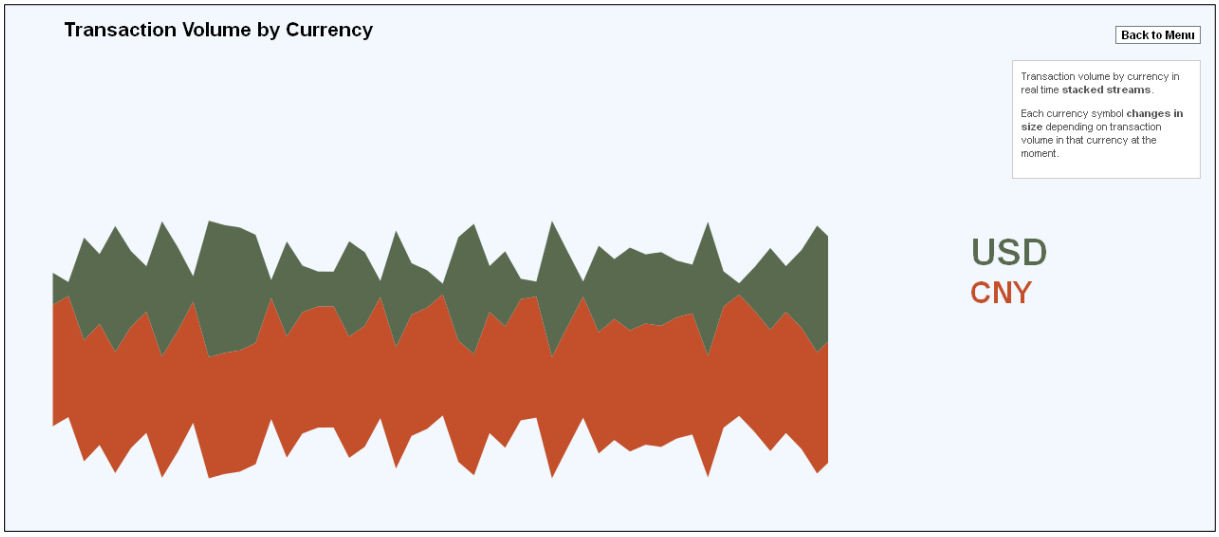
Here is visualization 7:



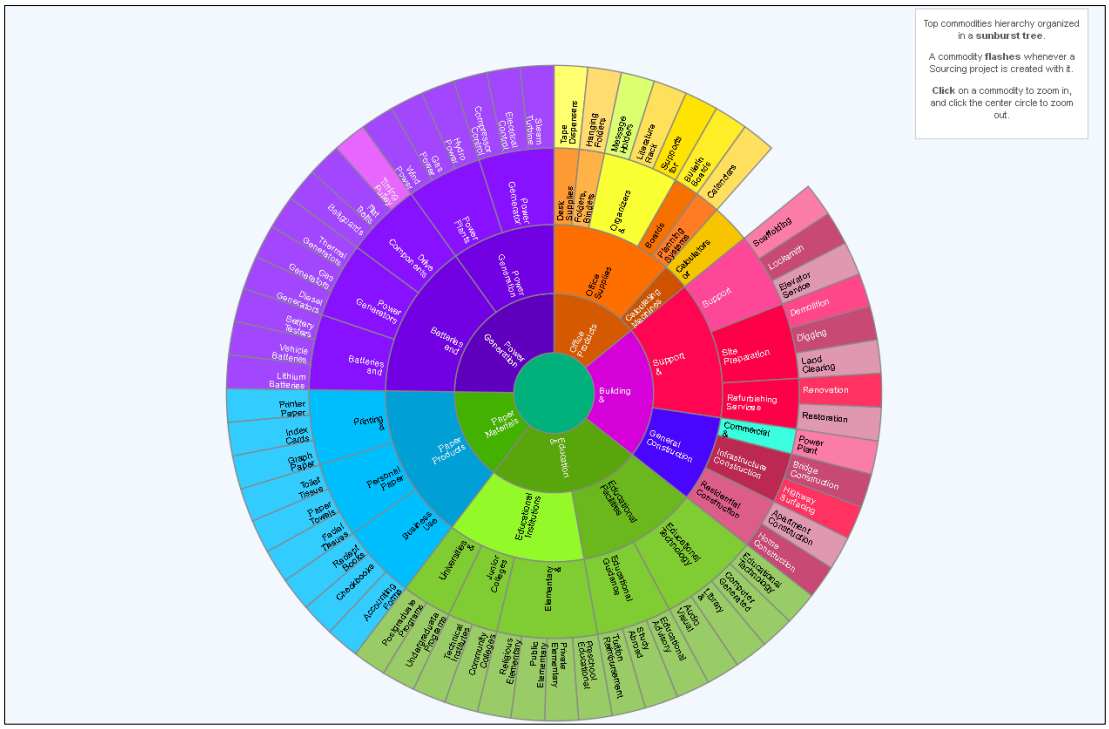
Here is visualization 8:



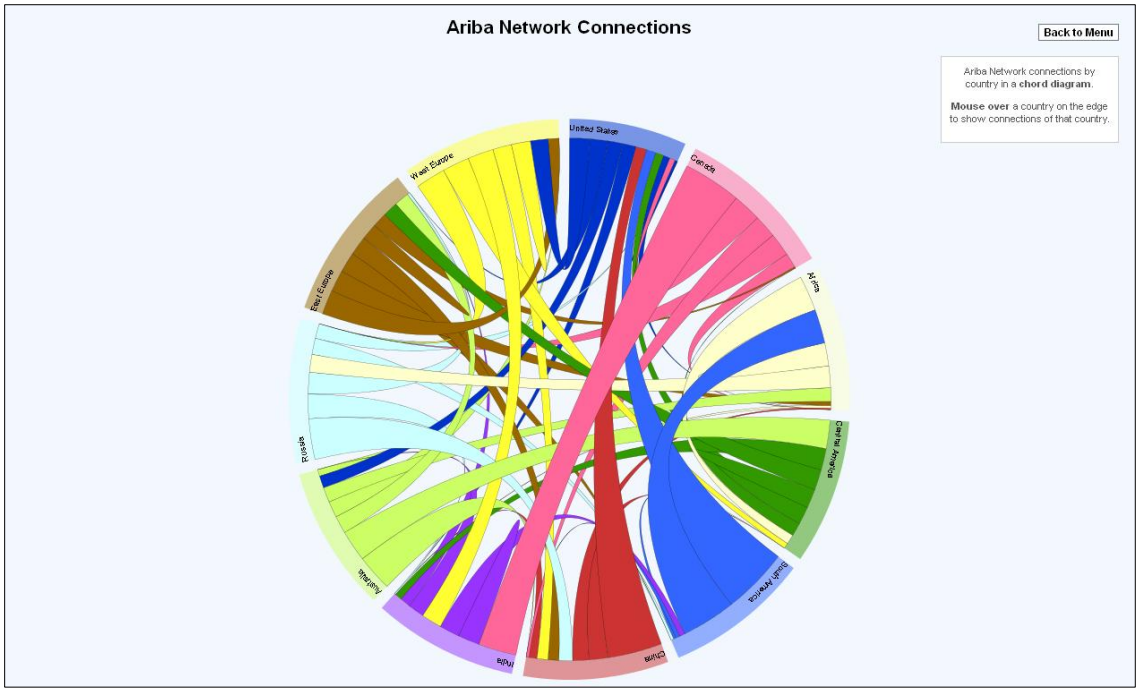
Here is visualization 9:



Here is visualization 10:



Here is visualization 11:



Here is visualization 12: