xbrl.us

ALTOVA

How To Download and Process SEC XBRL Data Directly from EDGAR

XBRL Technology Webinar Series

Alexander Falk, CEO, Altova, Inc.

XBRL eXtensible Business Reporting Language

xbrl.us

ALTOVA

Agenda

- Introduction
- Downloading all XBRL data from EDGAR
  - Accessing the SEC's EDGAR archive of all XBRL files via RSS feeds
  - Downloading the ZIP file enclosures for each filing
  - Special considerations for early years
- Organizing your downloaded files
  - You now have 105,840 ZIP files totaling 14.1 GB
  - Making them accessible by date, CIK, ticker
- Processing and validating the XBRL filings
- Extracting useful information, e.g., financial ratios

Introduction

- The SEC's EDGAR System holds a wealth of XBRL-formatted financial data for over 9,700 corporate entities.
- Accessing XBRL-formatted SEC data can seem like a daunting task and most people think it requires creating a database before the first data-point can be pulled.
- In this webinar we will show you how to pull all the data directly from the SEC archive and download it to your computer, then process it, and perform financial analysis.
- This will hopefully spark ideas that you can use in your own applications to take advantage of the computer-readable XBRL data now freely available.

For all source code examples we will be using Python 3.3.3, since it is widely available on all operating system platforms and also provides for easily readable code.

Obviously the approach shown in this webinar can easily be implemented in other languages, such as Java, C#, etc.

Why don't you just store it in a database?

- Processing and validating the XBRL files once and then storing the extracted data in a relational database is not necessarily a bad approach… BUT:
- Any data quality analysis as well as validation statistics and a documentation of data inconsistencies with respect to the calculation linkbase require the data to remain in XBRL format
- New XBRL technologies, like XBRL Formula and the XBRL Table Linkbase provide the potential for future applications that require the original data to be in XBRL format and not shredded and stored in a database
- Similarly the ability to query the data with an XQuery 3.0 processor only exists if the data remains in its original XML-based XBRL format

Clearly, if the only goal of processing the filings is to derive numerical data from the facts and build financial analytics, then pre-processing and shredding the data into a database may still be your best approach…

The XBRL.US database by Phillip Engel as well as Herm Fisher's talk at the last XBRL.US conference about using Arelle to populate a PostgreSQL database are good starting points for the database-based approach.

Downloading the data – accessing the RSS feeds

- In addition to the nice web-based user interface for searching XBRL filings, the SEC website offers RSS feeds to all XBRL filings ever received:
  - http://www.sec.gov/spotlight/xbrl/filings-and-feeds.shtml
- In particular, there is a monthly, historical archive of all filings with XBRL exhibits submitted to the SEC, beginning with the inception of the voluntary program in 2005:
  - http://www.sec.gov/Archives/edgar/monthly/
- This is our starting point, and it contains one RSS file per month from April 2005 until the current month of March 2014.

Downloading the data – understanding the RSS feed


```
<?xml version="1.0" encoding="windows-1252"?>
<?xml-stylesheet type="text/xsl" href="/rss/styles/shared_xsl_stylesheet_v2.xml"?>
<rss version="2.0">
  <channel>
    <title>All XBRL Data Submitted to the SEC for 2014-03</title>
    <link>http://www.sec.gov/spotlight/xbrl/filings-and-feeds.shtml</link>
    <atom:link href="http://www.sec.gov/Archives/edgar/monthly/xbrlrss-2014-03.xml" rel="self" type="application/rss+xml" xmlns
    <description>This is a list all of the filings containing XBRL for 2014-03</description>
    <language>en-us</language>
    <pubDate>Wed, 05 Mar 2014 00:00:00 EST</pubDate>
    <lastBuildDate>Wed, 05 Mar 2014 00:00:00 EST</lastBuildDate>
    <item>
      <title>COPART INC (0000900075) (Filer)</title>
      <link>http://www.sec.gov/Archives/edgar/data/900075/000114544314000288/0001145443-14-000288-index.htm</link>
      <guid>http://www.sec.gov/Archives/edgar/data/900075/000114544314000288/0001145443-14-000288-xbrl.zip</guid>
      <enclosure url="http://www.sec.gov/Archives/edgar/data/900075/000114544314000288/0001145443-14-000288-xbrl.zip" length="1
      <description>10-Q</description>
      <pubDate>Wed, 05 Mar 2014 17:28:10 EST</pubDate>
      <edgar:xbrlFiling xmlns:edgar="http://www.sec.gov/Archives/edgar">
        <edgar:companyName>COPART INC</edgar:companyName>
        <edgar:formType>10-Q</edgar:formType>
        <edgar:filingDate>03/05/2014</edgar:filingDate>
        <edgar:cikNumber>0000900075</edgar:cikNumber>
        <edgar:accessionNumber>0001145443-14-000288</edgar:accessionNumber>
        <edgar:fileNumber>000-23255</edgar:fileNumber>
        <edgar:acceptanceDatetime>20140305172810</edgar:acceptanceDatetime>
        <edgar:period>20140131</edgar:period>
        <edgar:assistantDirector>2</edgar:assistantDirector>
        <edgar:assignedSic>5500</edgar:assignedSic>
        <edgar:fiscalYearEnd>0731</edgar:fiscalYearEnd>
```

Downloading the data – loading the RSS feed


```
def SECdownload(year, month):
    root = None
    feedFile = None
    feedData = None
    good_read = False
    itemIndex = 0
    edgarFilingsFeed = 'http://www.sec.gov/Archives/edgar/monthly/xbrlrss-' + str(year) + '-' + str(month).zfill(2) + '.xml'
    print( edgarFilingsFeed )
    if not os.path.exists( "sec/" + str(year) ):
        os.makedirs( "sec/" + str(year) )
    if not os.path.exists( "sec/" + str(year) + '/' + str(month).zfill(2) ):
        os.makedirs( "sec/" + str(year) + '/' + str(month).zfill(2) )
    target_dir = "sec/" + str(year) + '/' + str(month).zfill(2) + '/'
    try:
        feedFile = urlopen( edgarFilingsFeed )
        try:
            feedData = feedFile.read()
            good_read = True
        finally:
            feedFile.close()
    except HTTPError as e:
        print( "HTTP Error:", e.code )
```

In Python we can easily use the urlopen function from the urllib package to open a file from a URL and read its data

Downloading the data – parsing the RSS feed to extract the ZIP file enclosure filename

```
# Process RSS feed and walk through all items contained
for item in feed.entries:
    print( item[ "summary" ], item[ "title" ], item[ "published" ] )
    try:
        # Identify ZIP file enclosure, if available
        enclosures = [ l for l in item[ "links" ] if l[ "rel" ] == "enclosure" ]
        if ( len( enclosures ) > 0 ):
            # ZIP file enclosure exists, so we can just download the ZIP file
            enclosure = enclosures[0]
            sourceurl = enclosure[ "href" ]
            cik = item[ "edgar_ciknumber" ]
            targetfname = target_dir+cik+'-'+sourceurl.split('/')[-1]
            retry_counter = 3
            while retry_counter > 0:
                good_read = downloadfile( sourceurl, targetfname )
                if good_read:
                    break
                else:
                    print( "Retrying:", retry_counter )
                    retry_counter -= 1
```

In Python we can easily use the feedparser 5.1.3 package to parse the RSS feed and extract the ZIP file name and CIK#

https://pypi.python.org/pypi/feedparser

Please note that for the local filename I am constructing that by inserting the CIK in front of the actual ZIP file name.

Downloading the data – loading the ZIP file enclosure

```
def downloadfile( sourceurl, targetfname ):
    mem_file = ""
    good_read = False
    xbrlfile = None
    if os.path.isfile( targetfname ):
        print( "Local copy already exists" )
        return True
    else:
        print( "Downloading:", sourceurl )
        try:
            xbrlfile = urlopen( sourceurl )
            try:
                mem_file = xbrlfile.read()
                good_read = True
            finally:
                xbrlfile.close()
        except HTTPError as e:
            print( "HTTP Error:", e.code )
        except URLError as e:
            print( "URL Error:", e.reason )
        except TimeoutError as e:
            print( "Timeout Error:", e.reason )
        except socket.timeout:
            print( "Socket Timeout Error" )
        if good_read:
            output = open( targetfname, 'wb' )
            output.write( mem_file )
            output.close()
        return good_read
```

Please note that it is prudent to first check if we already have a local copy of that particular filing. We should only download the ZIP file, if we don't find it on our local machine yet.

Also, please note that common courtesy dictates that if you plan to download all years 2005-2014 of XBRL filings from the SEC's EDGAR archive, you should do so during off-peak hours, e.g. night-time or weekends in order not to tax the servers during normal business hours, as you will be downloading 105,840 files or 14.1GB of data!

Downloading the data – special considerations for early years

- For years 2005-2007 most filings do not yet contain a ZIP file enclosure.
- Even in 2008-2009 some filings can occasionally be found that are not yet provided in a ZIP file.
- If you are interested in analyzing data from those early years, a little bit of extra work is required to download all the individual XBRL files from a filing and then ZIP them up locally.
- If done properly, all future analysis can then access all the filings in the same manner directly from the ZIP files.

Downloading the early years – ZIPping the XBRL files on our local machine

```
# We need to manually download all XBRL files here and ZIP them ourselves...
linkname = item[ "link" ].split('/')[-1]
linkbase = os.path.splitext(linkname)[0]
cik = item[ "edgar_ciknumber" ]
zipfname = target_dir+cik+'-'+linkbase+"-xbrl.zip"
if os.path.isfile( zipfname ):
    print( "Local copy already exists" )
else:
    edgarNamespace = {'edgar': 'http://www.sec.gov/Archives/edgar'}
    currentItem = list(root.iter( "item" ))[itemIndex]
    xbrlFiling = currentItem.find( "edgar:xbrlFiling", edgarNamespace )
    xbrlFilesItem = xbrlFiling.find( "edgar:xbrlFiles", edgarNamespace )
    xbrlFiles = xbrlFilesItem.findall( "edgar:xbrlFile", edgarNamespace )
    if not os.path.exists(  target_dir+"temp" ):
        os.makedirs( target_dir+"temp" )
    zf = zipfile.ZipFile( zipfname, "w" )
    try:
        for xf in xbrlFiles:
            xfurl = xf.get( "{http://www.sec.gov/Archives/edgar}url" )
            if xfurl.endswith( (".xml",".xsd") ):
                targetfname = target_dir+"temp/"+xfurl.split('/')[-1]
                retry_counter = 3
                while retry_counter > 0:
                    good_read = downloadfile( xfurl, targetfname )
                    if good_read:
                        break
                    else:
                        print( "Retrying:", retry_counter )
                        retry_counter -= 1
                zf.write( targetfname, xfurl.split('/')[-1], zipfile.ZIP_DEFLATED )
                os.remove( targetfname )
    finally:
        zf.close()
        os.rmdir( target_dir+"temp" )
```

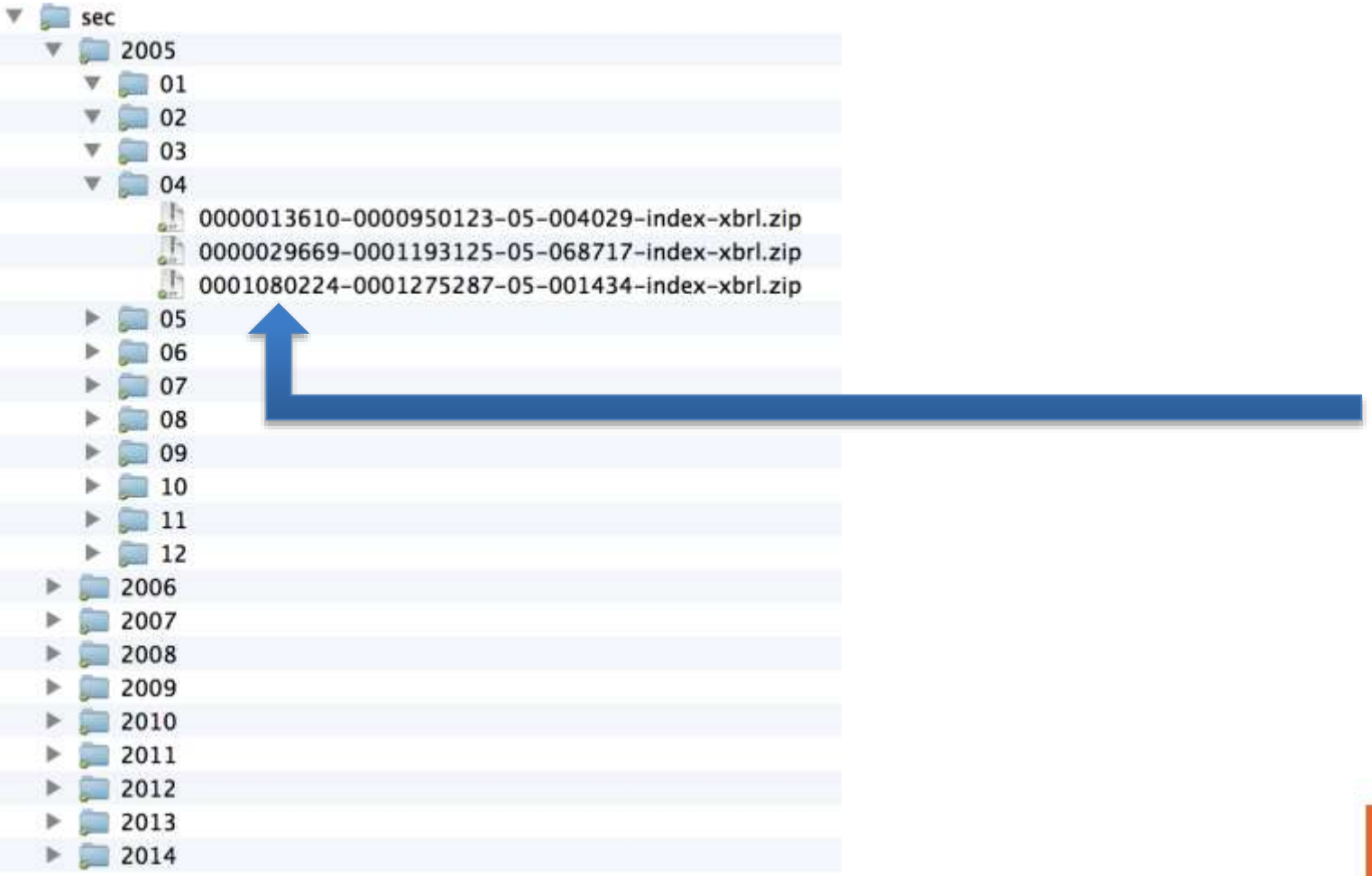
If we want to download data from the early years, we need to use two additional Python packages:

(a) The ElementTree XML parser, because feedparser cannot handle multiple nested elements for the individual filings
(b) The zipfile package so that we can ZIP the downloaded files up ourselves

Demo time

```
Last login: Fri Mar  7 07:15:45 on ttys000
MillenniumFalcon:~ alf$ cdpyp
MillenniumFalcon:PyProjects alf$ py loadSECfilings.py -y 2014 -m 3
http://www.sec.gov/Archives/edgar/monthly/xbrlrss-2014-03.xml
All XBRL Data Submitted to the SEC for 2014-03
10-K FARMERS CAPITAL BANK CORP (0000713095) (Filer) Fri, 07 Mar 2014 17:29:24 EST
Downloading: http://www.sec.gov/Archives/edgar/data/713095/000143774914003694/0001437749-14-003694-xbrl.zip
----------
10-K KBS Real Estate Investment Trust, Inc. (0001330622) (Filer) Fri, 07 Mar 2014 17:29:22 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1330622/000133062214000004/0001330622-14-000004-xbrl.zip
----------
10-K NOODLES & Co (0001275158) (Filer) Fri, 07 Mar 2014 17:29:11 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1275158/000127515814000011/0001275158-14-000011-xbrl.zip
----------
10-K TG THERAPEUTICS, INC. (0001001316) (Filer) Fri, 07 Mar 2014 17:29:09 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1001316/000114420414014207/0001144204-14-014207-xbrl.zip
----------
10-K Athlon Energy Inc. (0001574648) (Filer) Fri, 07 Mar 2014 17:28:53 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1574648/000104746914001993/0001047469-14-001993-xbrl.zip
----------
10-K ServisFirst Bancshares, Inc. (0001430723) (Filer) Fri, 07 Mar 2014 17:27:57 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1430723/000114420414014205/0001144204-14-014205-xbrl.zip
----------
497 FRANKLIN INVESTORS SECURITIES TRUST (0000809707) (Filer) Fri, 07 Mar 2014 17:27:25 EST
Downloading: http://www.sec.gov/Archives/edgar/data/809707/000137949114000270/0001379491-14-000270-xbrl.zip
----------
10-K NPC Restaurant Holdings, LLC (0001548621) (Filer) Fri, 07 Mar 2014 17:25:32 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1548621/000154862114000008/0001548621-14-000008-xbrl.zip
----------
S-1/A Owlhead Minerals Corp. (0001578523) (Filer) Fri, 07 Mar 2014 17:25:04 EST
Downloading: http://www.sec.gov/Archives/edgar/data/1578523/000149315214000664/0001493152-14-000664-xbrl.zip
```

Organizing the downloaded files – file system structure



For my purposes I have already organized the files at the same time as I have downloaded them. Since the RSS feeds group the filings nicely by year and month, I have created one subdirectory for each year and one subdirectory for each month.

In order to easily process the filings of one particular reporting entity, I have also inserted the CIK# in front of the ZIP file name, since the SEC-assigned accession number does not really help me when I want to locate all filings for one particular filer.

Organizing the downloaded files – making them accessible by date, CIK, ticker

```
def lookup_cik(ticker, name=None):
    # Given a ticker symbol, retrieves the CIK.
    good_read = False
    ticker = ticker.strip().upper()
    url = 'http://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&CIK={cik}&count=10&output=xml'.format(cik=ticker)

    try:
        xmlFile = urlopen( url )
        try:
            xmlData = xmlFile.read()
            good_read = True
        finally:
            xmlFile.close()
    except HTTPError as e:
        print( "HTTP Error:", e.code )
    except URLError as e:
        print( "URL Error:", e.reason )
    except TimeoutError as e:
        print( "Timeout Error:", e.reason )
    except socket.timeout:
        print( "Socket Timeout Error" )
    if not good_read:
        print( "Unable to lookup CIK for ticker:", ticker )
        return
    try:
        root = ET.fromstring(xmlData)
    except ET.ParseError as perr:
        print( "XML Parser Error:", perr )

    try:
        cikElement = list(root.iter( "CIK" ))[0]
        return int(cikElement.text)
    except StopIteration:
        pass
```

Selecting files for further processing by date is trivial due to our directory structure. Similarly, selecting filings by CIK is easily facilitated since the filenames of all filings now begin with the CIK.

The only part that needs a little work is to make them accessible by ticker – fortunately the SEC provides a web service interface to look up company information and filings by ticker, and the resulting XML also contains the CIK, which we can retrieve via simple XML parsing.

Processing and validating the XBRL filings

- Now that we have all the data organized, we can use Python to process, e.g., all filings from one filer for a selected date range
- For this webinar we are going to use RaptorXML® Server to process and validate the XBRL filings
- RaptorXML can directly process the filings inside of ZIP files, so no manual extraction step is necessary
- We can also pass an entire batch of jobs to RaptorXML at once:

```
result = call(["raptorxmlxbrl", "xbrl", "--listfile", joblist] )
```

- We can do this either via direct call as shown here, or over the HTTP API provided by RaptorXML Server.

Shameless Plug:

RaptorXML® is built from the ground up to be optimized for the latest standards and parallel computing environments. Designed to be highly cross-platform capable, the engine takes advantage of today's ubiquitous multi-CPU computers to deliver lightning fast processing of XML and XBRL data.

Therefore, we can pass an entire batch of jobs to RaptorXML to process and validate in parallel, maximizing CPU utilization and available system resources.

Processing and validating the filings – building the job list based on dates and CIK

```
def appendjoblist( year, month, cik=None ):
    target_dir = "sec/" + str(year) + '/' + str(month).zfill(2) + '/'
    cikPattern = None
    if not cik==None:
        cikStr = list(map( str, cik ))
        cikPattern = tuple(cs.zfill(10) for cs in cikStr)
    try:
        for filename in os.listdir( target_dir ):
            add_file = False
            if os.path.splitext(filename)[1] == ".zip":
                if cik == None:
                    add_file = True
                else:
                    if filename.startswith( cikPattern ):
                        add_file = True
                if add_file:
                    zipname = target_dir+filename
                    joblist.append( zipname )
    except FileNotFoundError as fe:
        print( 'Error: no SEC filings found in directory', target_dir )
```

This directory contains all filings for one particular year and month.

We iterate over all the files in that directory.

If a list of CIKs was provided, then we make sure the filename starts with the CIK.

xbrl.us

ALTOVA®

Demo time – validating all 2010-2014 filings for ORCL

XBRL eXtensible Business Reporting Language

```
MillenniumFalcon:PyProjects alf$ py valSECfilings.py -f 2010 -t 2014 -k ORCL
Tickers:  ['ORCL']
CIKs:  [1341439]
16 validation jobs queued up for RaptorXML+XBRL
Running batch of 16 validation jobs...
file:///Users/alf/Dropbox/PyProjects/sec/2010/03/0001341439-0001193125-10-070192-xbrl.zip%7Czip/orcl-20100228.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/07/0001341439-0001193125-10-151896-xbrl.zip%7Czip/orcl-20100531.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/09/0001341439-0001193125-10-219697-xbrl.zip%7Czip/orcl-20100831.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/12/0001341439-0001193125-10-285703-xbrl.zip%7Czip/orcl-20101130.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2011/03/0001341439-0001193125-11-081178-xbrl.zip%7Czip/orcl-20110228.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2011/06/0001341439-0001193125-11-174819-xbrl.zip%7Czip/orcl-20110531.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2011/09/0001341439-0001193125-11-255436-xbrl.zip%7Czip/orcl-20110831.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2011/12/0001341439-0001193125-11-351954-xbrl.zip%7Czip/orcl-20111130.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2012/03/0001341439-0001193125-12-129918-xbrl.zip%7Czip/orcl-20120229.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2012/06/0001341439-0001193125-12-284007-xbrl.zip%7Czip/orcl-20120531.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2012/09/0001341439-0001193125-12-401697-xbrl.zip%7Czip/orcl-20120831.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2012/12/0001341439-0001193125-12-513009-xbrl.zip%7Czip/orcl-20121130.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2013/03/0001341439-0001193125-13-122271-xbrl.zip%7Czip/orcl-20130228.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2013/06/0001341439-0001193125-13-272832-xbrl.zip%7Czip/orcl-20130531.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2013/09/0001341439-0001193125-13-373455-xbrl.zip%7Czip/orcl-20130831.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2013/12/0001341439-0001193125-13-481081-xbrl.zip%7Czip/orcl-20131130.xml: result="OK"
Elapsed real time: 3.541717052459717 seconds
```

Demo time – validating all 2010-2014 filings for AAPL

XBRL eXtensible Business Reporting Language

```
MillenniumFalcon:PyProjects alf$ py valSECfilings.py -f 2010 -t 2014 -k AAPL
Tickers:  ['AAPL']
CIKs:  [320193]
19 validation jobs queued up for RaptorXML+XBRL
Running batch of 19 validation jobs...
file:///Users/alf/Dropbox/PyProjects/sec/2010/01/0000320193-0001193125-10-012085-xbrl.zip%7Czip/aapl-20091226.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/01/0000320193-0001193125-10-012091-xbrl.zip%7Czip/aapl-20090926.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/04/0000320193-0001193125-10-088957-xbrl.zip%7Czip/aapl-20100327.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/07/0000320193-0001193125-10-162840-xbrl.zip%7Czip/aapl-20100626.xml: result="OK"
file:///Users/alf/Dropbox/PyProjects/sec/2010/10/0000320193-0001193125-10-238044-xbrl.zip%7Czip/aapl-20100925.xml: result="OK"
Inconsistency: file:///Users/alf/Dropbox/PyProjects/sec/2010/10/0000320193-0001193125-10-238044-xbrl.zip%7Czip/aapl-20100925.xml:23861: Rounded value 5105000000
 of summation item <us-gaap:GrossProfit> is not equal to 12207000000.
  Reason: it is not equal to the sum of the rounded and weighted values of its contributing items.
    <us-gaap:SalesRevenueNet> ('decimals'='-6') = 12207000000, weighted with 1 = 12207000000
  Details:
    calculationArc-summation-item-inconsistent: Rounded value 5105000000 of summation item <us-gaap:GrossProfit> is not equal to 12207000000.
Inconsistency: file:///Users/alf/Dropbox/PyProjects/sec/2010/10/0000320193-0001193125-10-238044-xbrl.zip%7Czip/aapl-20100925.xml:23854: Rounded value 6136000000
 of summation item <us-gaap:GrossProfit> is not equal to 15700000000.
  Reason: it is not equal to the sum of the rounded and weighted values of its contributing items.
    <us-gaap:SalesRevenueNet> ('decimals'='-6') = 15700000000, weighted with 1 = 15700000000
  Details:
    calculationArc-summation-item-inconsistent: Rounded value 6136000000 of summation item <us-gaap:GrossProfit> is not equal to 15700000000.
Inconsistency: file:///Users/alf/Dropbox/PyProjects/sec/2010/10/0000320193-0001193125-10-238044-xbrl.zip%7Czip/aapl-20100925.xml:23849: Rounded value 5625000000
 of summation item <us-gaap:GrossProfit> is not equal to 13499000000.
  Reason: it is not equal to the sum of the rounded and weighted values of its contributing items.
    <us-gaap:SalesRevenueNet> ('decimals'='-6') = 13499000000, weighted with 1 = 13499000000
  Details:
    calculationArc-summation-item-inconsistent: Rounded value 5625000000 of summation item <us-gaap:GrossProfit> is not equal to 13499000000.
Inconsistency: file:///Users/alf/Dropbox/PyProjects/sec/2010/10/0000320193-0001193125-10-238044-xbrl.zip%7Czip/aapl-20100925.xml:23844: Rounded value 6411000000
 of summation item <us-gaap:GrossProfit> is not equal to 15683000000.
  Reason: it is not equal to the sum of the rounded and weighted values of its contributing items.
    <us-gaap:SalesRevenueNet> ('decimals'='-6') = 15683000000, weighted with 1 = 15683000000
  Details:
    calculationArc-summation-item-inconsistent: Rounded value 6411000000 of summation item <us-gaap:GrossProfit> is not equal to 15683000000.
```

Extracting useful information, e.g. financial ratios

- While it is interesting to discuss data quality of XBRL filings, and to muse over inconsistencies for some filers or whether the SEC should put more stringent validation checks on the data it accepts, we really want to do more here…
- Can we extract useful financial ratios from these XBRL filings?
- For example, from the balance sheet:

$$Current\ Ratio = \frac{Current\ Assets}{Current\ Liabilites}$$

$$Quick\ Ratio = \frac{\text{Cash} + \text{Short-Term Marketable Securities} + \text{Accounts Receivable}}{Current\ Liabilites}$$

$$Cash\ Ratio = \frac{\text{Cash} + \text{Short-Term Marketable Securities}}{Current\ Liabilites}$$

Extracting useful information – passing Python script to the built-in Python interpreter inside RaptorXML

- We can ask RaptorXML to execute some Python script code if the XBRL validation has succeeded.
- From our outer Python code we pass the script to RaptorXML:

```
result = call(["raptorxmlxbrl", "xbrl", "--script="+script,
               "--listfile", joblist] )
```

- Then, whenever validation succeeds, RaptorXML will execute that script using its built-in Python interpreter:

```
def on_xbrl_valid( job, instance ):
```

RaptorXML is written in C++ and available on all major operating system platforms, including Windows, Linux, MacOS, etc.

To facilitate easy customization and building powerful solutions on top of RaptorXML, it includes a built-in Python interpreter that makes the entire DTS, XBRL instance, schema, and other relevant information accessible to 3rd party developers.

Extracting useful information – calculating ratios using Python script inside RaptorXML

- The RaptorXML Python API makes available all necessary components of the XBRL instance document and the DTS (=Discoverable Taxonomy Set)
- To make the code more easily understandable for this webinar, we've created a few helper functions to locate relevant facts and print them, e.g., for the Current Ratio:

```
# Current Ratio
currentRatio = 0
print( "\t\tCurrent Ratio = Current Assets / Current Liabilities:" )
currentAssetsFacts = factFinder( instance, fasb_ns, "AssetsCurrent" )
currentLiabilitiesFacts = factFinder( instance, fasb_ns, "LiabilitiesCurrent" )
currentAssets = printFacts( currentAssetsFacts, 3, docEndDate )
currentLiabilities = printFacts( currentLiabilitiesFacts, 3, docEndDate )
if not currentLiabilities==0:
    currentRatio = currentAssets / currentLiabilities
print( 3 * "\t", "Current Ratio = ".ljust(100-3*8), '{0:.2f}'.format( currentRatio ) )
```

Extracting useful information – not all financial ratios are easy to calculate

- Not all ratios are calculated from elements that can be easily found in the US GAAP taxonomy
- Even for those ratios where an exact match exists in the taxonomy, it is often necessary to walk through the calculation linkbase chain and identify appropriate matches in order to calculate ratios across filings from different entities
- For further information please see Roger Debreceny, et al: "Feeding the Information Value Chain: Deriving Analytical Ratios from XBRL filings to the SEC", Draft research paper, December 2010:http://eycarat.faculty.ku.edu//myssi/_pdf/2-Debreceny-XBRL%20Ratios%2020101213.pdf

Extracting useful information – Quick Ratio

- One such example is the Cash fact needed to calculate the Quick Ratio: we have to try three different facts depending on what is available in the actual XBRL filing:

```
# Quick Ratio
quickRatio = 0
print( "\t\tQuick Ratio = ( Cash + Short-Term Marketable Securities + Accounts Receivable ) / Current Liabilities:" )
cashFacts = factFinder( instance, fasb_ns, "Cash" )
if len(cashFacts)==0:
    cashFacts = factFinder( instance, fasb_ns, "CashAndCashEquivalentsAtCarryingValue" )
if len(cashFacts)==0:
    cashFacts = factFinder( instance, fasb_ns, "CashCashEquivalentsAndShortTermInvestments" )
marketableSecuritiesFacts = factFinder( instance, fasb_ns, "MarketableSecuritiesCurrent" )
if len(marketableSecuritiesFacts)==0:
    marketableSecuritiesFacts = factFinder( instance, fasb_ns, "AvailableForSaleSecuritiesCurrent" )
accountsReceivableFacts = factFinder( instance, fasb_ns, "AccountsReceivableNetCurrent" )
currentLiabilitiesFacts = factFinder( instance, fasb_ns, "LiabilitiesCurrent" )
cash = printFacts( cashFacts, 3, docEndDate )
marketableSecurities = printFacts( marketableSecuritiesFacts, 3, docEndDate )
accountsReceivable = printFacts( accountsReceivableFacts, 3, docEndDate )
currentLiabilities = printFacts( currentLiabilitiesFacts, 3, docEndDate )
if not currentLiabilities==0:
    quickRatio = ( cash + marketableSecurities + accountsReceivable ) / currentLiabilities
print( 3 * "\t", "Quick Ratio = ".ljust(100-3*8), '{0:.2f}'.format( quickRatio ) )
```

xbrl.us

ALTOVA®

Demo time – calculating financial ratios for some companies in my investment portfolio

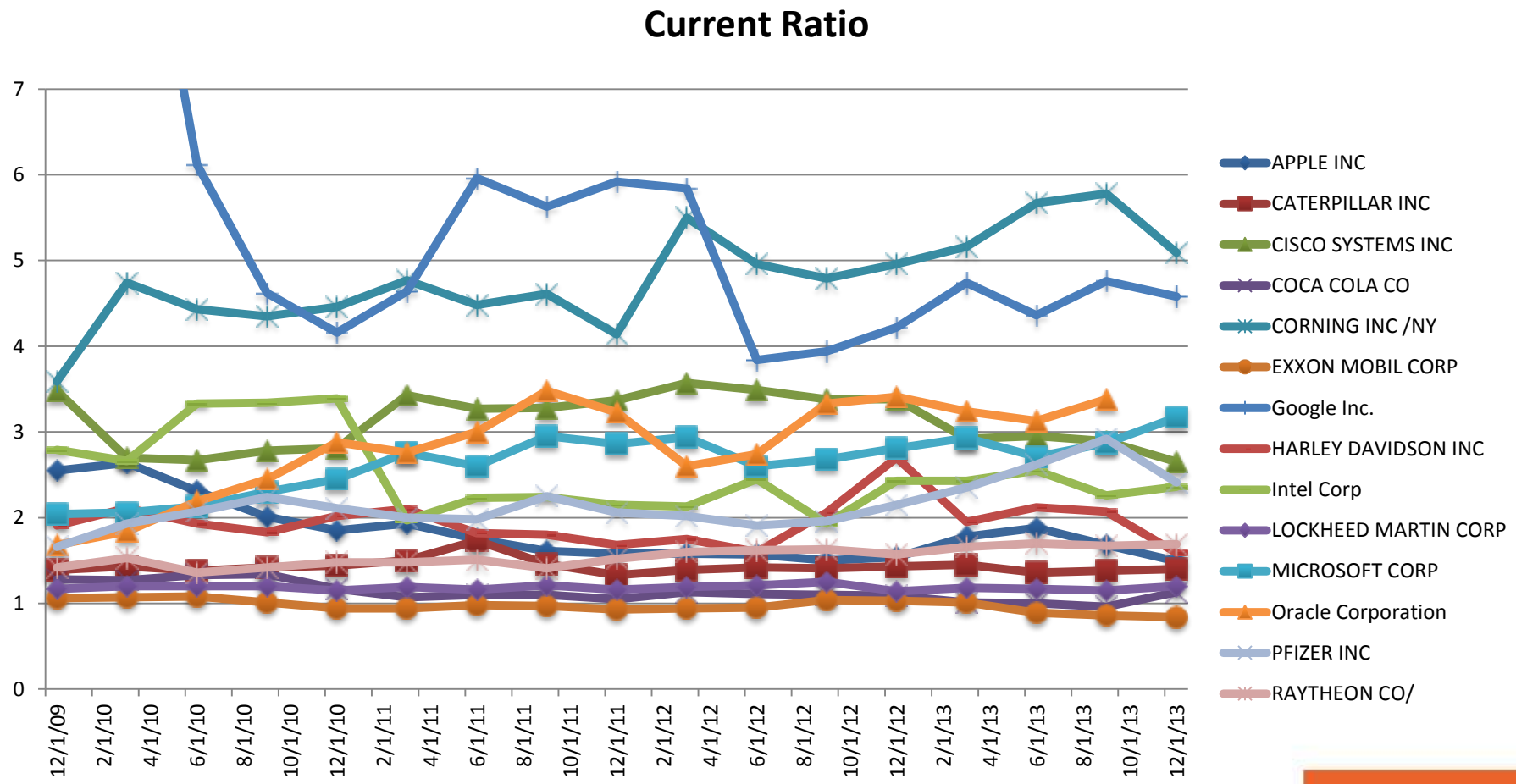
XBRL eXtensible Business Reporting Language

```
MillenniumFalcon:PyProjects alf$ py valSECfilings.py -f 2010 -t 2014 -k AAPL,CAT,CSCO,GLW,GOOG,HOG,INTC,KO,LMT,MSFT,ORCL,PFE,RTN,XOM -s extractRatios.py
Tickers:  ['AAPL', 'CAT', 'CSCO', 'GLW', 'GOOG', 'HOG', 'INTC', 'KO', 'LMT', 'MSFT', 'ORCL', 'PFE', 'RTN', 'XOM']
CIKs:  [320193, 18230, 858877, 24741, 1288776, 793952, 50863, 21344, 936468, 789019, 1341439, 78003, 1047122, 34088]
244 validation jobs queued up for RaptorXML+XBRL
Partition: 1
Running batch of 20 validation jobs...
Document and Entity Information:
        Document Type                                                   10-Q
        Entity Registrant Name                                          APPLE INC
        Entity Central Index Key                                        0000320193
        Document Period End Date                                        2010-03-27
        Document Fiscal Period Focus                                    Q2
        Document Fiscal Year Focus                                      2010
Analytical Ratios:
       Balance Sheet:
              Current Ratio = Current Assets / Current Liabilities:
                      Assets Current                                    $   32,336,000,000
                      Liabilities Current                               $   12,229,000,000
                      Current Ratio =                                   2.64
              Quick Ratio = ( Cash + Short-Term Marketable Securities + Accounts Receivable ) / Current Liabilities:
                      Cash And Cash Equivalents At Carrying Value       $   10,018,000,000
                      Available For Sale Securities Current             $   13,137,000,000
                      Accounts Receivable Net Current                   $    2,886,000,000
                      Liabilities Current                               $   12,229,000,000
                      Quick Ratio =                                     2.13
              Cash Ratio = ( Cash + Short-Term Marketable Securities ) / Current Liabilities:
                      Cash And Cash Equivalents At Carrying Value       $   10,018,000,000
                      Available For Sale Securities Current             $   13,137,000,000
                      Liabilities Current                               $   12,229,000,000
                      Cash Ratio =                                      1.89
Document and Entity Information:
        Document Type                                                   10-Q
        Entity Registrant Name                                          LOCKHEED MARTIN CORP
        Entity Central Index Key                                        0000936468
        Document Period End Date                                        2010-03-28
        Document Fiscal Period Focus                                    Q1
        Document Fiscal Year Focus                                      2010
Analytical Ratios:
       Balance Sheet:
              Current Ratio = Current Assets / Current Liabilities:
                      Assets Current                                    $   13,919,000,000
                      Liabilities Current                               $   11,572,000,000
                      Current Ratio =                                   1.20
```

In this example, we are calculating the Current Ratio, Quick Ratio, and Cash Ratio for a set of companies that happen to be part of my investment portfolio…

In addition to printing the ratios on the screen, we also store them in a CSV file for further processing and graphing…

Financial Ratio Results



Q&A and next steps

- Thank you for your time and for watching this webinar! Time for some Q&A now…

- For more information on the XBRL data available from the SEC, please visit http://xbrl.sec.gov/
- We also plan to post the Python scripts shown here on GitHub in the near future.
- If you would like to learn more about RaptorXML®, please visit the Altova website at http://www.altova.com/raptorxml.html
- For all other Altova® XBRL solutions, including taxonomy development, data mapping, and rendering tools, please visit http://www.altova.com/solutions/xbrl.html
- Free 30-day evaluation versions of all Altova products can be downloaded from http://www.altova.com/download.html