UnitedHealth Group

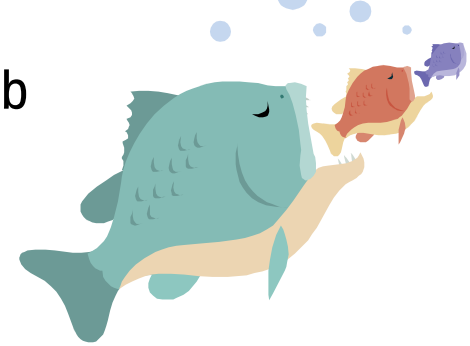
Implementing Asynchronous Triggers Using LogMiner Loader

Using the Loader to integrate Rdb data with data from service providers

Russ Remple, UnitedHealth Group

Oracle Rdb Technical Forum, 10/20/2009

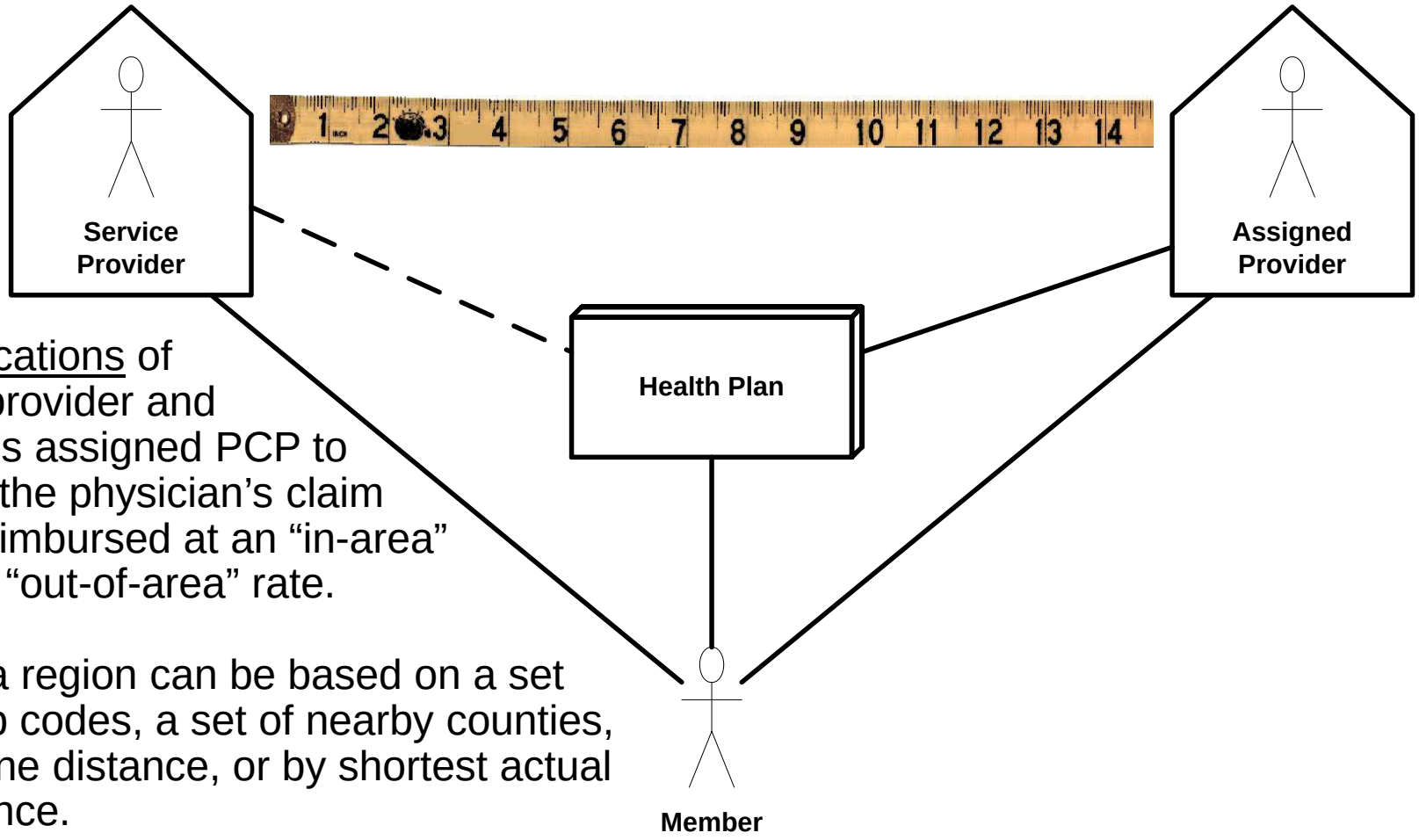
History



- Brief history of UnitedHeath Group with OpenVMS/Rdb
  - UnitedHeath Group acquired PacifiCare Health Systems in 2006
  - PacifiCare's primary HMO system, servicing about 2 million members, is called "NICE"
  - NICE is an OpenVMS/COBOL/Rdb-based application built in the late 80's
- UnitedHeath Group history with LogMiner
  - In 1999-2000, we started using Rdb LogMiner for Tuxedo-based data replication of 15 tables – this project was successful… but kind of *kludgey*
  - In 2004-2005 we implemented JCC LogMiner Loader for continuous replication to Oracle DB (ORAC on Linux) of about 50 tables – this was *very* successful, it is still in use, very popular, and is still growing
  - So in 2008-2009 we did something a bit different involving two source tables and a project to replace an outdated Geographic Information Systems (GIS)…

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Business Context: Why would we use GIS?



- We use locations of the service provider and the member's assigned PCP to determine if the physician's claim should be reimbursed at an "in-area" rate or at an "out-of-area" rate.

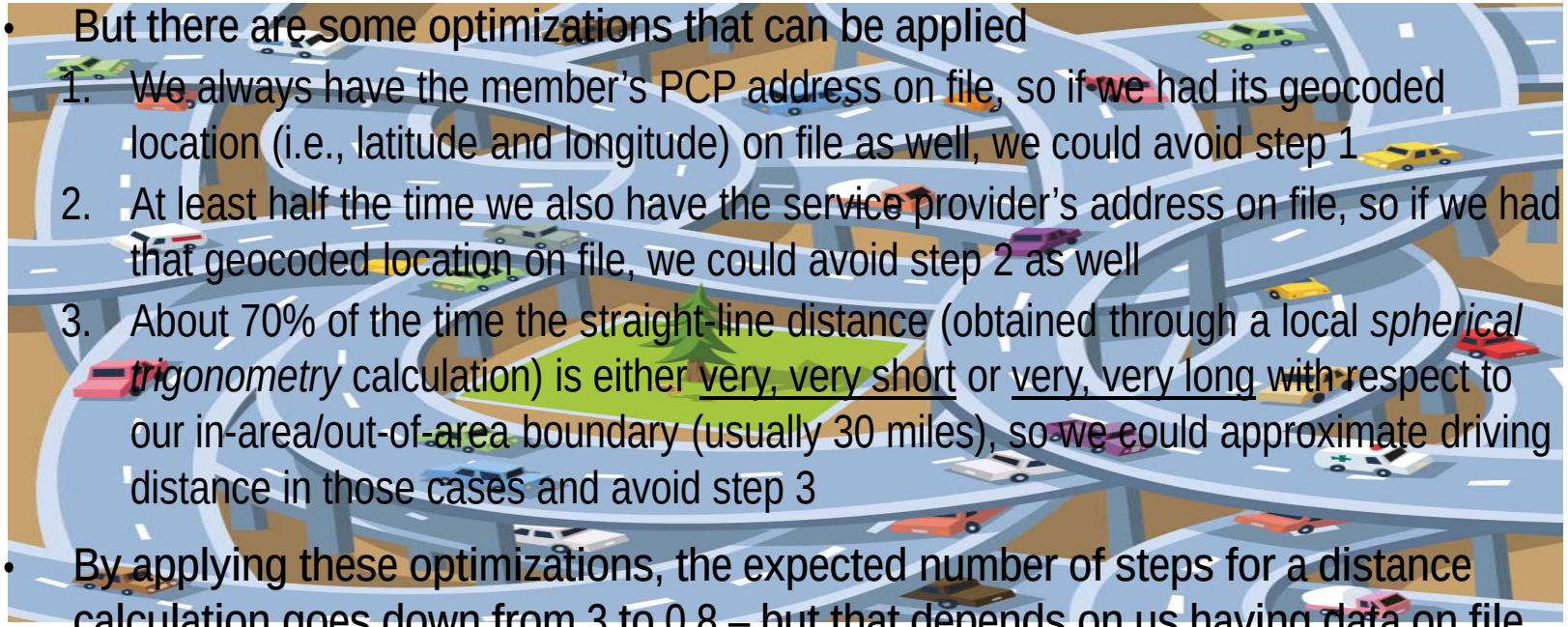
- The in-area region can be based on a set of nearby zip codes, a set of nearby counties, by straight-line distance, or by shortest actual driving distance.

- Driving distance is the most popular way… and also the most complicated.

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Business Context: How GIS is applied

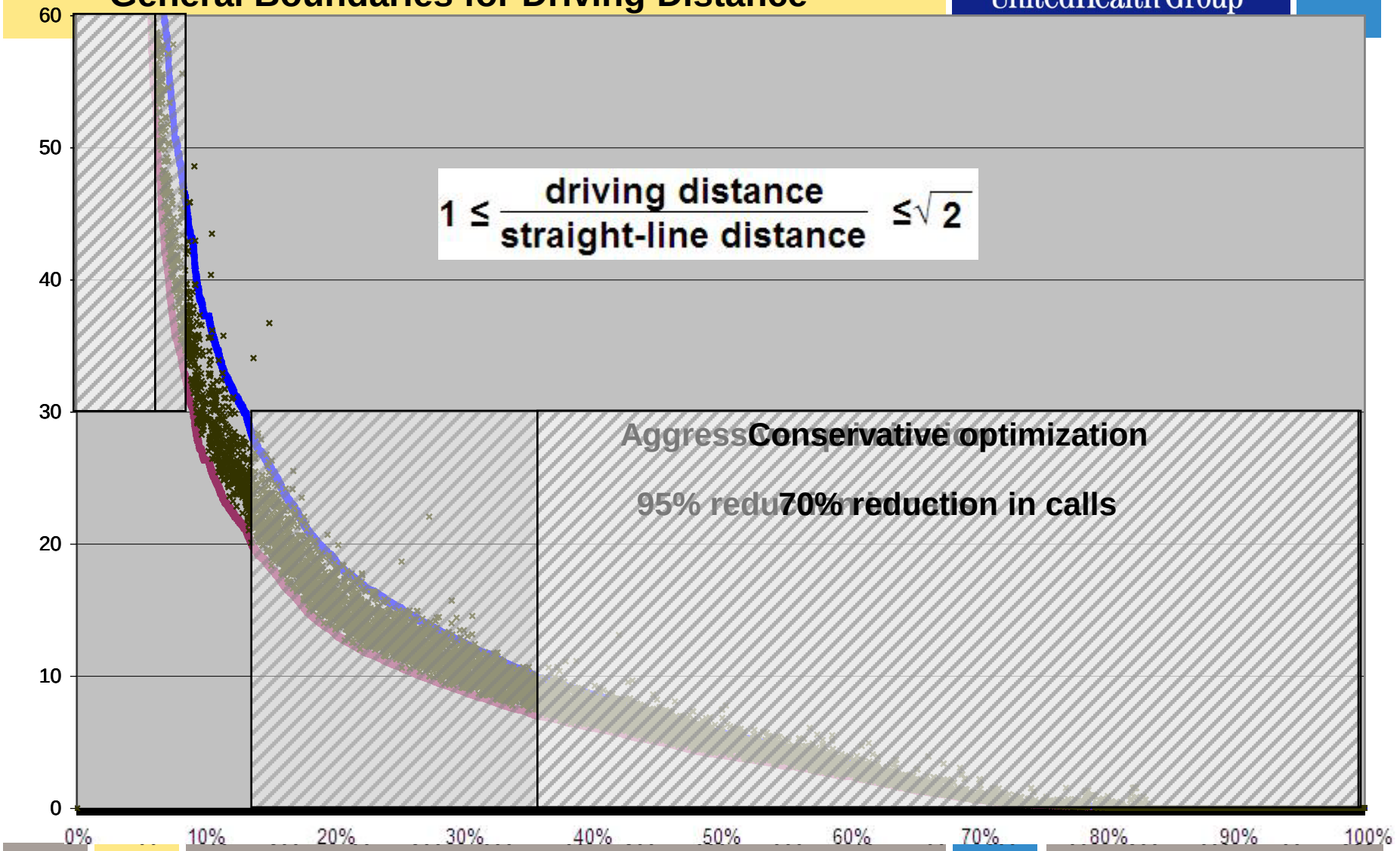
- There are three steps involved in making a in-area/out-of-area determination based on driving distance
  1. Geocode the "from" address (i.e., that of the member's assigned provider/PCP)
  2. Geocode the "to" address (i.e., that of the service provider)
  3. Calculate the actual driving distance from one geocoded location to the other



- But there are some optimizations that can be applied
  1. We always have the member's PCP address on file, so if we had its geocoded location (i.e., latitude and longitude) on file as well, we could avoid step 1
  2. At least half the time we also have the service provider's address on file, so if we had that geocoded location on file, we could avoid step 2 as well
  3. About 70% of the time the straight-line distance (obtained through a local *spherical trigonometry* calculation) is either very, very short or very, very long with respect to our in-area/out-of-area boundary (usually 30 miles), so we could approximate driving distance in those cases and avoid step 3
- By applying these optimizations, the expected number of steps for a distance calculation goes down from 3 to 0.8 – but that depends on us having data on file

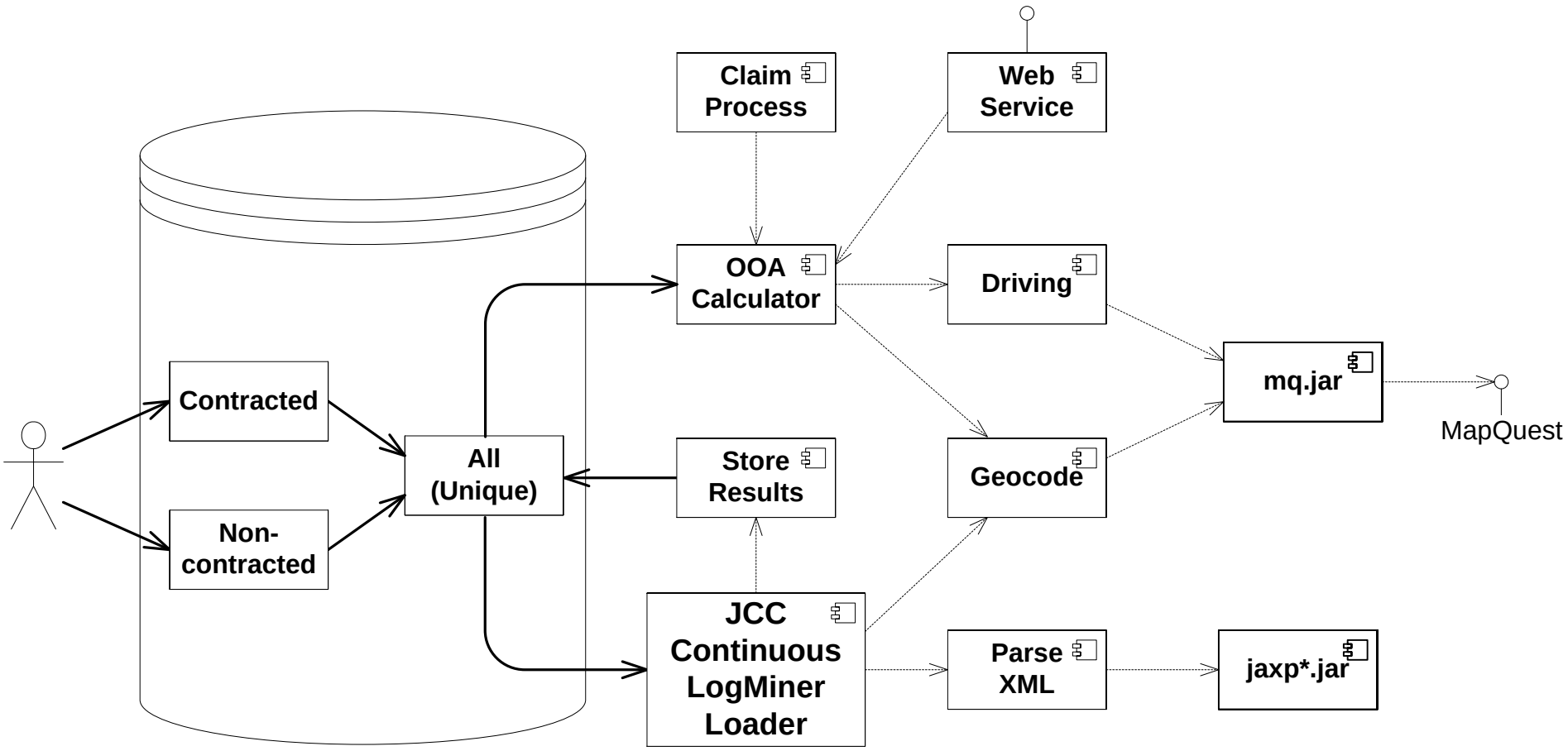
Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

General Boundaries for Driving Distance



Any use, copying or distribution without written permi

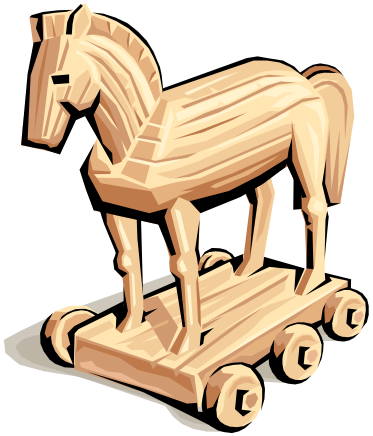
Technical Context: Why would we use JCC LML?



Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Context: strategies/challenges

- Strategy
  - Use LogMiner Loader to process addresses as they are updated
  - Store the results back into the Rdb database
  - Leverage the experience and the utilities developed for integrating 3GL and Java programs on OpenVMS Alpha using JNI to do things in Java that are "easier" to do there than in other languages (e.g., XML processing)
  - Minimize the use of C, maximize Java and COBOL based on local skills
  - Leverage existing ACMS and WSIT tools for exposing our own logic to the intranet

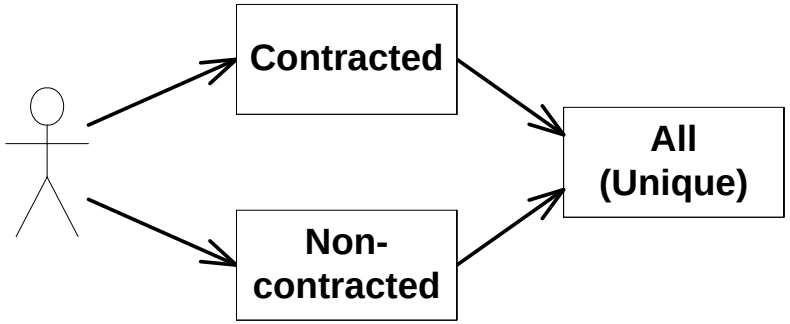


- Challenges
  - Managing unique addresses in a way that performs well and is maintainable
  - Use LogMiner Loader API/XML target to capture and process new/changed addresses
  - Parsing and processing LML XML data effectively and efficiently
  - Integrating with geocoding and distance calculation service APIs (MapQuest)
  - Integrating with the claims process
  - Providing intranet access to GIS information

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Manage unique addresses

- Challenge
  - There are many duplicate addresses, some address types are not relevant, and addresses are stored in multiple tables
  - We need to use triggers to solve for this, and since triggers execute synchronously, there may be an impact on performance
  - Avoid “crazy SQL” and keep the solution maintainable
- Solution
  - Create a new table for GEOCODED_ADDRESSES and populate using triggers
  - Only do inserts/updates synchronously and defer deletes to the LogMiner Loader
  - Leverage a stored procedure from multiple triggers to isolate shared logic (e.g., managing reference counts)



Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Manage unique addresses

- Example trigger source:

```
before delete on NON_CONTRACTED_PROVIDER referencing old as O (
      call GEOCODED_ADDRESS_CHANGE (O.STREET_LINE_1, ..., -1)
   ) for each row

after insert  on NON_CONTRACTED_PROVIDER referencing new as N (
      call GEOCODED_ADDRESS_CHANGE (N.STREET_LINE_1, ..., +1)
   ) for each row

after update of STREET_LINE_1, CITY, STATE, ZIP_CDE
   on NON_CONTRACTED_PROVIDER referencing old as O new as N (
      call GEOCODED_ADDRESS_CHANGE (O.STREET_LINE_1, ..., -1)
      call GEOCODED_ADDRESS_CHANGE (N.STREET_LINE_1, ..., +1)
   ) for each row
```

- Similar triggers are on the table that is the source for contracted provider address information

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Manage unique addresses

- Example stored procedure source:

```
declare :existing_REFERENCE_COUNT INTEGER;
declare :existing_GEOCODE_STATUS char(2);

select coalesce(sum(REFERENCE_COUNT),0), coalesce(max(GEOCODE_STATUS),'')
   into :existing_REFERENCE_COUNT, :existing_GEOCODE_STATUS
   from GEOCODED_ADDRESS where KEY_…=:in_KEY_…;

IF :existing_REFERENCE_COUNT > 0 THEN
    IF :existing_REFERENCE_COUNT + :in_REFERENCE_DELTA > 0 THEN
        update GEOCODED_ADDRESS
           set REFERENCE_COUNT = REFERENCE_COUNT + :in_REFERENCE_DELTA
           where KEY_…=:in_KEY_…;
     ELSE
        update GEOCODED_ADDRESS set REFERENCE_COUNT = 0, GEOCODE_STATUS = 'DE'
           where KEY_…=:in_KEY_…;
    END IF;

 ELSE IF :in_REFERENCE_DELTA > 0 THEN
    IF :existing_GEOCODE_STATUS = '' THEN
        insert into GEOCODED_ADDRESS
           (KEY_…,     REFERENCE_COUNT,     FIRST_REFERENCE,    GEOCODE_STATUS) values
           (:in_KEY_…, :in_REFERENCE_DELTA, :in_REFERENCE_NAME, 'NE');
     ELSE
        update GEOCODED_ADDRESS
           set REFERENCE_COUNT = :in_REFERENCE_DELTA, GEOCODE_STATUS = 'NE'
           where KEY_…=:in_KEY_…;
    END IF;
END IF; END IF;
```


Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Unique addresses in GEOCODED_ADDRESS

KEY_STREET	**CHAR(30)**	**Street Address**
KEY_CITY	**CHAR(20)**	**City Name**
KEY_STATE	**CHAR(2)**	**State Code**
KEY_ZIP_5	**CHAR(5)**	**5 Digit Zip Code**
KEY_ZIP_4	**CHAR(4)**	**4-digit Zip extension**
GEOCODE_STATUS	CHAR(2)	NE - new address added to the table, RE - reprocess an address, DE - delete - no longer used, CP - address has been geocoded, ER - do not geocode address
CLEAN_STREET	CHAR(30)	Cleaned up Street Address
CLEAN_CITY	CHAR(20)	Cleaned up City Name
CLEAN_STATE	CHAR(2)	Cleaned up State Code
CLEAN_ZIP_5	CHAR(5)	Cleaned 5 digit ZIP code
CLEAN_ZIP_4	CHAR(4)	Cleaned up 4-digit Zip extension
GEOCODE_COUNTY	CHAR(50)	County returned by MapQuest
GEOCODE_LATITUDE	REAL	Latitude coordinate returned by MapQuest
GEOCODE_LONGITUDE	REAL	Longitude coordinate returned by MapQuest
GEOCODE_PRECISION	CHAR(20)	Result code returned by vendor
GEOCODE_VENDOR	CHAR(2)	Vendor used, i.e., MQ for MapQuest.
GEOCODE_DATE	DATE VMS	Date/Time address was geocoded
REFERENCE_COUNT	INTEGER	Number of records that have this address
ADDRESS_STATUS_CODE	CHAR(1)	Translated status from GEOCODE_PRECISION
FIRST_REFERENCE	CHAR(8)	Identifies provider address type

Key source table values

Status driving LogMiner Loader process

Information returned by GIS (MapQuest)

Housekeeping

LogMiner Loader example filter:

```
FilterMap~GEOCODED_ADDRESS~( \
          GEOCODE_STATUS in ('NE', 'RE', 'DE') \
     )
```

LogMiner Loader filter to ignore ‘CP’ and ‘ER’

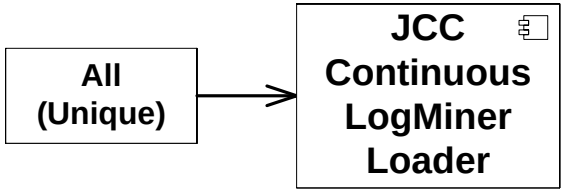
Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Using LML API/XML target

- Challenge
  - Use LogMiner Loader API/XML target for the first time
  - There may be “timing issues” when LogMiner Loader falls behind
- Solution
  - We leveraged all our existing code for running loaders
  - Constructing the INI files was not difficult – followed the examples and documentation

- LogMiner Loader example .INI file entries

```
checkpoint~1~lml_internal~ASYNCH
parallel~4~4~constrained
output~API~synch~SY_1315_BU_01~TRANSACTION~XML
API~CONNECT~msgConnect
API~SEND~msgSend~fixed args…
API~DISCONNECT~msgDisconnect
```



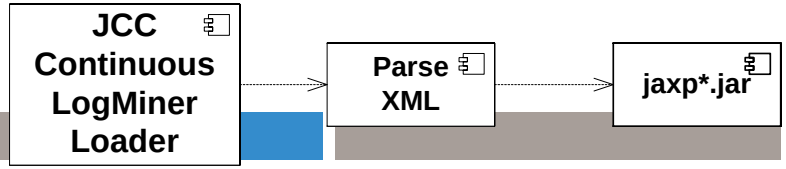
- Example C code in SY_1315_BU_01

```
int MSGCONNECT(void **handle_ptr, char *ip_address,
               char *ip_port, char *topic_name,
               char *lml_name, int timeout) { … }
int MSGSEND(void **handle_ptr, int msgSize, char *msg) { … }
int MSGDISCONNECT(void **handle_ptr) { … }
```


Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Passing/parsing XML

- Challenge
  - XML documents can be large since they contain all the rows updated in a transaction, so DOM-style parsing may be too memory-intensive.
  - However, using the SAX parser can be a bit tedious…
  - Leveraging Java requires the C string passed in from the JCC LML API be “wrapped” somehow, so there is potential for additional memory inefficiencies if done incorrectly
- Solution
  - Use the Streaming API for XML (STaX) parser in the Java 6 implementation of JAXP (available in Java 5 too in the Sun reference implementation) – this is a high-performance, low-memory footprint parser that isn’t too difficult to use
  - Use JNI “DirectByteBuffer” to pass C string to Java as a ByteBuffer
  - Wrap the ByteBuffer as an InputStream, which can be used in JAXP to create an XMLEventReader directly – no memory is copied!
  - Use iterator symatics “hasNext()” and “next()” to pull each row of an Rdb transaction from the XML document – parsing is done step by step as data is processed



Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Passing/parsing XML

- Example C code – creating a byte buffer and setting the value in the request class:

```
status = JNI_CreateJavaVM(&(env->jVM), (void **)&jEnv, &jVMArgs);
…
jobject byteBuffer = (*jEnv)->NewDirectByteBuffer(jEnv, msg, msgSize);

status = getJniMethodId(jEnv, class, "setJccXmlByteBuffer",
            "(Ljava/nio/ByteBuffer;)V", &setJccXmlByteBufferMid));
(*jEnv)->CallVoidMethod(jEnv, object,
                        setJccXmlByteBufferMid, byteBuffer);
(*jEnv)->DeleteLocalRef(jEnv, byteBuffer);
```

//not shown, but don't forget to check JNI status after method calls too!

- Related example Java code – setting the byte buffer in the request:

```
public class GeocodeJccXmlParseRequest {
    private ByteBuffer jccXmlByteBuffer;

    public void setJccXmlByteBuffer(ByteBuffer jccXmlByteBuffer) {
        this.jccXmlByteBuffer = jccXmlByteBuffer;
    }
```


Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Passing/parsing XML

- A bit more example Java code – actually using the byte buffer requires the creation of an InputStream:

```
public InputStream getJccXmlInputStream() {
    return new InputStream() {

        public synchronized int read() throws IOException {
            return jccXmlByteBuffer.hasRemaining()
                    ? jccXmlByteBuffer.get() : -1;
        }

        public synchronized int read(byte[] byteArray, int offset,
                                     int requestSize)
                                throws IOException {
            int responseSize = Math.min(requestSize,
                                        jccXmlByteBuffer.remaining());
            jccXmlByteBuffer.get(byteArray, offset, responseSize);
            return responseSize == 0 ? -1 : responseSize;
        }
    };
}
```


Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Passing/parsing XML

- Last bit of example Java code – setting up the XML event reader using the InputStream for the byte buffer (note that this, like other **Service* classes in NICE, follow a "Command" pattern):

```
public class GeocodeJccXmlParseService {
    private XMLInputFactory xmlReaderFactory;

    public GeocodeJccXmlParseService(String initString) {
        xmlReaderFactory = XMLInputFactory.newInstance();
        xmlReaderFactory.setProperty(XMLInputFactory.IS_COALESCING,
                                     Boolean.TRUE);
    }

    public GeocodeJccXmlParseResponse execute
                (GeocodeJccXmlParseRequest request) throws Exception {
        return new GeocodeJccXmlParseResponse(
            xmlReaderFactory.createXMLEventReader(
                request.getJccXmlInputStream()));
    }
```


Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

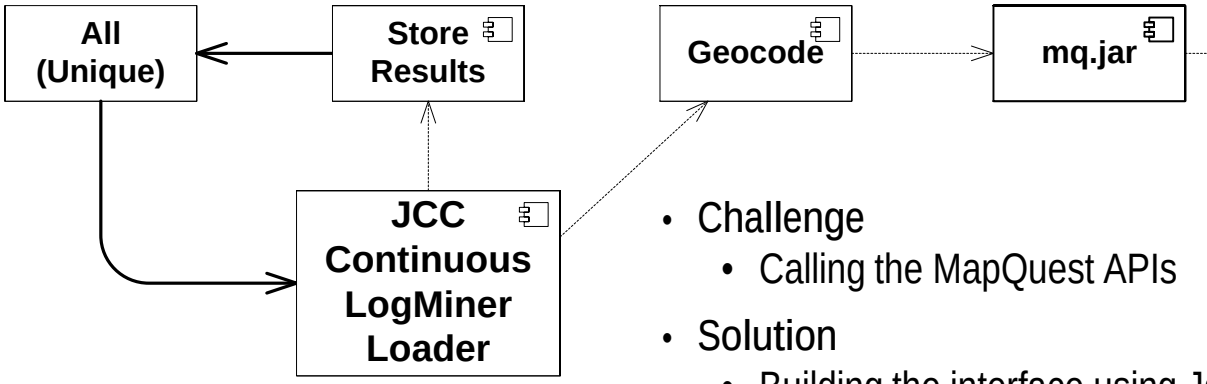
Technical Challenge: Passing/parsing XML

- Last bit of related example C code – now we can iterate through each row in the database transaction:

```
status = getJniMethodId(jEnv, class, "hasNext", "()Z", &hasNextMid);
status = getJniMethodId(jEnv, class, "next",
          "()Lcom/uhg/nice/client/geocode/GeocodeAddressRequest;",
          &nextMid);

int jitem;
jboolean hasNext = (*jEnv)->CallBooleanMethod(jEnv, object, hasNextMid);
for (jitem=0; hasNext; jitem++) {
  jobject addressObject = (*jEnv)->CallObjectMethod(jEnv,
                          object, nextMid, jitem);

  status = geocodeJccXmlLoadHandleAddress(jEnv, addressObject);

  (*jEnv)->DeleteLocalRef(jEnv, addressObject);
  hasNext = (*jEnv)->CallBooleanMethod(jEnv, object, hasNextMid);
}
```


Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Geocoding addresses



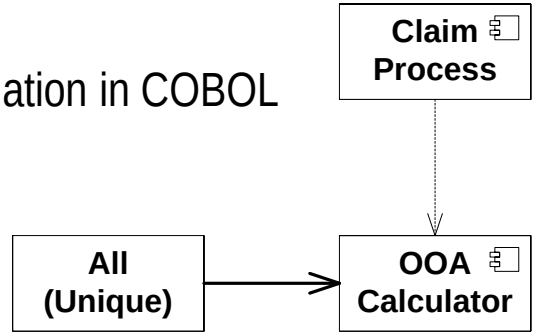
- Challenge
  - Calling the MapQuest APIs
- Solution
  - Building the interface using Java API was not difficult – followed the examples and documentation, as well as leveraged MapQuest's very helpful developer support (and lots of testing!)

```
address.setStreet(request.getKeyStreet()); …
geocodeClient.geocode(address, lcOriginResults);
gaOrigin = (GeoAddress) lcOriginResults.getAt(0);
response.setGeocodeLatitude(
     Double.toString(gaOrigin.getLatLng().getLatitude()));
response.setGeocodeLongitude(
     Double.toString(gaOrigin.getLatLng().getLongitude()));
response.setCleanStreet(gaOrigin.getStreet());…
```

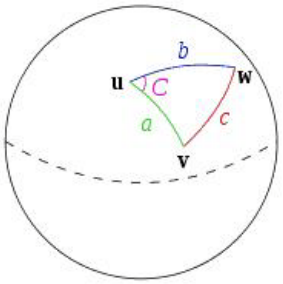

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Straight-line distance

- Challenge
  - So geocodes are “just numbers” – how do you get a real distance?
- Solution
  - Easy – use the Spherical Law of Cosines. We did the calculation in COBOL



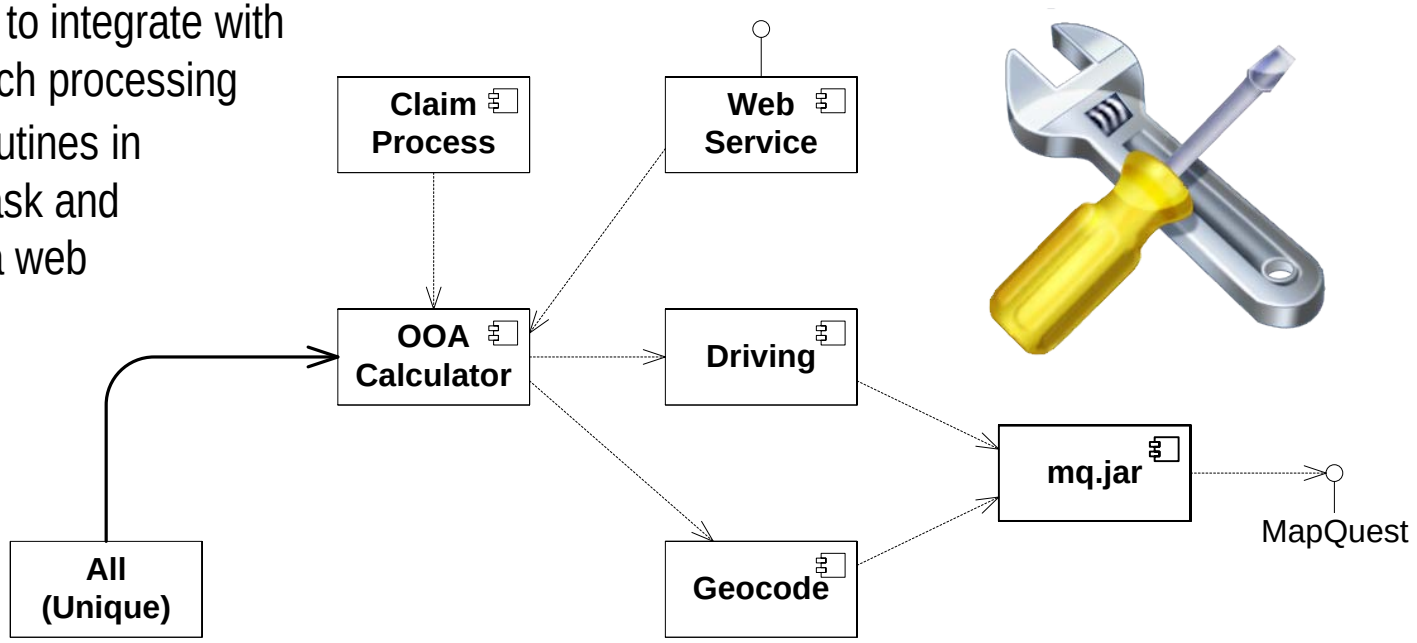

```
01 WS_CONSTANTS.
   05  PI             COMP-1 value 3.14159265.
   05  PLANET_RADIUS COMP-1 value 3959.
 . . .
COMPUTE WS_STRAIGHT_DISTANCE =
    PLANET_RADIUS * function acos(
        (function sin(LATITUDE  OF WS_FROM_DEGREES * PI / 180) *
         function sin(LATITUDE  OF WS_TO_DEGREES   * PI / 180)) +
        (function cos(LATITUDE  OF WS_FROM_DEGREES * PI / 180) *
         function cos(LATITUDE  OF WS_TO_DEGREES   * PI / 180) *
         function cos(LONGITUDE OF WS_TO_DEGREES   * PI / 180 –
                      LONGITUDE OF WS_FROM_DEGREES * PI / 180)))
```



Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Actually using the data

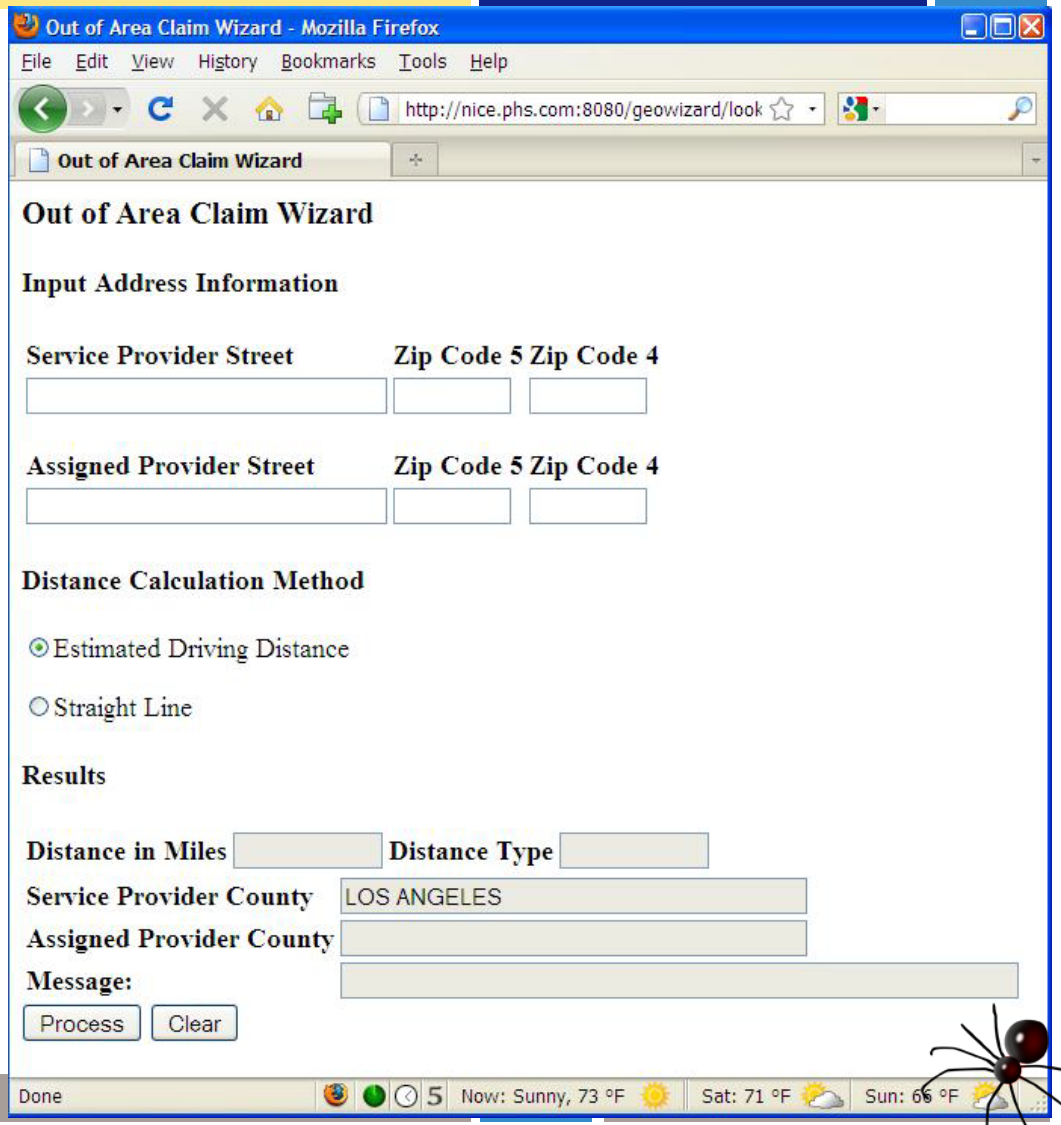
- Challenge
  - Geocode synchronously only when needed (apply optimizations discussed earlier)
  - Expose information to batch and online processes
- Solution
  - Use similar Java interface to do driving calculations (similar API), create COBOL subroutines to integrate with COBOL batch processing
  - Wrap subroutines in an ACMS task and expose as a web service



Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Technical Challenge: Actually using the data

- Built a simple, one-page web application for internal business users to leverage these services
- Replaces an existing desktop application supporting hundreds of casual users (total volume is less than 1,000 transactions per day)
- Gives answers that are consistent with claims process
- Can run on any Java server platform… but we decided to go with OpenVMS Itanium because it was convenient



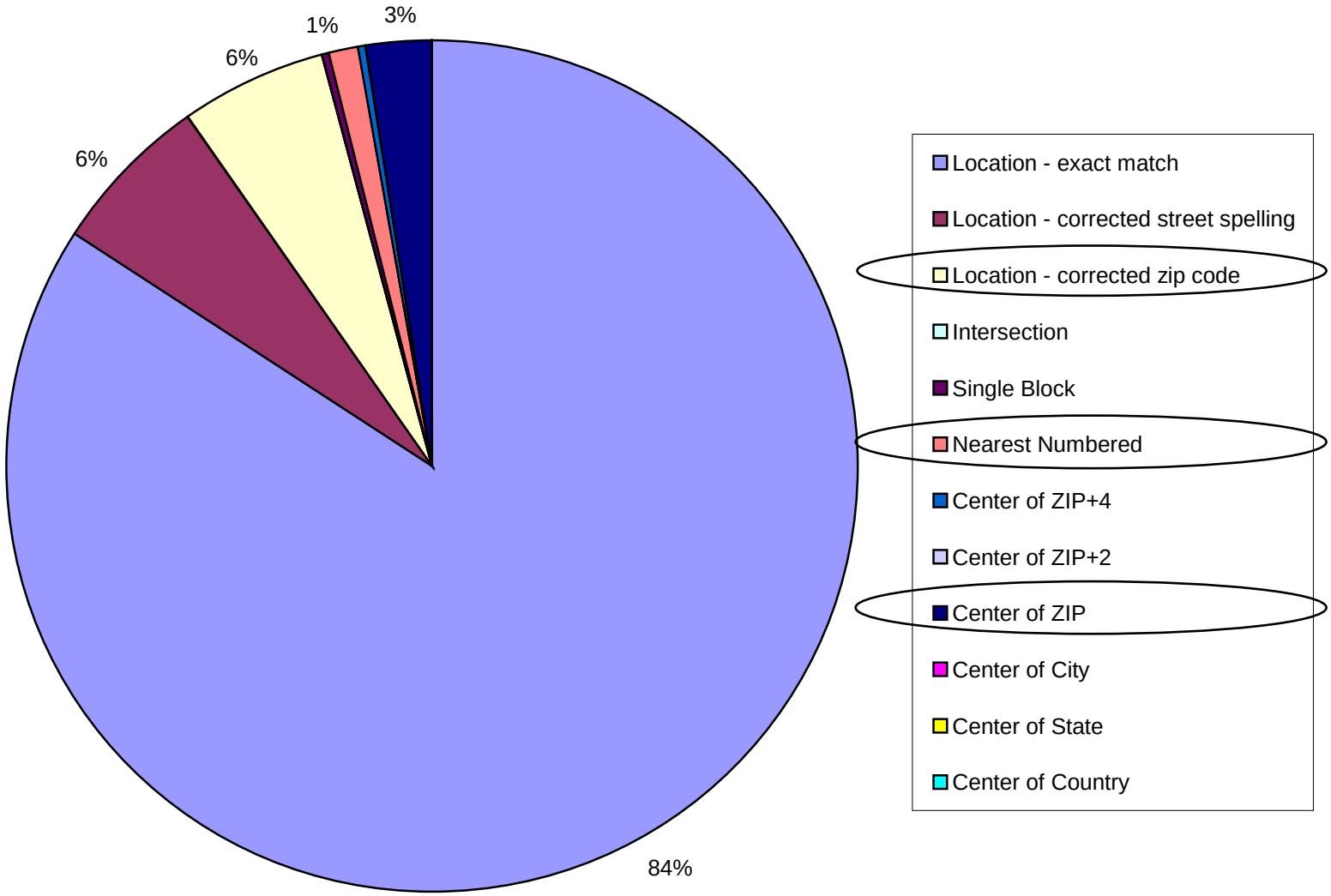
Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Business Benefits

- The new GIS solution should have a lower annual cost than the outdated one did
- The new solution always uses up-to-date information, and it is easy to reprocess any addresses that are suspect (using the 'RE' status value) – note that the actual longitude/latitude values of a correctly located address should never go out of date, but the "clean address" information might, so some regular reprocessing may be performed.
- Clean address information can be used to correct incorrect source values (e.g., when clean address zip code does not match source zip – it may have been changed by USPS!)
- The resulting database of clean addresses can be leveraged in other processing (e.g., it is standardized, it includes zip+4 information that we may not otherwise have, etc)
- Status values can assist the business in addressing data quality in new addresses even before the first claim is received, which may reduce payment issues for new providers due to setup issues
- Status values can also be used to evaluate general data quality of our existing database, which may generally improve business processes such as claims payment

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

Business Benefits: Data Quality Analysis



Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.

UnitedHealth Group

Lessons Learned

- We did a lot of prototyping and that definitely helped. We made sure the project was doable using this approach before we discussed it with the business. If you decide to do something like this, give yourself some time to experiment a little before diving into a full-on project commitment.
- JCC LogMiner Loader uses subprocesses for parallel processing. Java uses huge amounts of BYTELM. Because subprocesses share this quota, our use of Java limited the number of parallel processes we could run. (This was important only at implementation when we geocoded our entire database of addresses.)
- We evaluated other GIS vendors and they were all a bit different, and not just with respect to “terms of use”. So you should test a lot before you decide. For example, one vendor was so biased toward “points of interest” that it would incorrectly place addresses on Vatican Lane in Dallas, TX as being in Italy! (This has since been fixed.)
- MapQuest API services are metered, but they only meter production use. (Some vendors meter all use, some don’t meter at all.) This allowed us to benchmark our critical processes and tune them without undue throttling. Performance and availability were both outstanding, and MapQuest’s support organization was very helpful.
- Using JNI from C can be a bit tricky. If you aren’t careful about deleting local references you can wind up with nasty memory leaks that are very difficult to track down. Most JNI examples are for calling C from Java (not the other way around), and a lot of examples out there are not coded carefully, so beware!
- We did not include any metering features on our side and I think we should have. Maybe in release 2?
- It is almost depressing to finish a project like this, because it was so much fun and it is too bad that it has to end. But it is nice to see it all working in production, and nice to talk about it with you all. Thank you.

Any use, copying or distribution without written permission from UnitedHealth Group is prohibited.