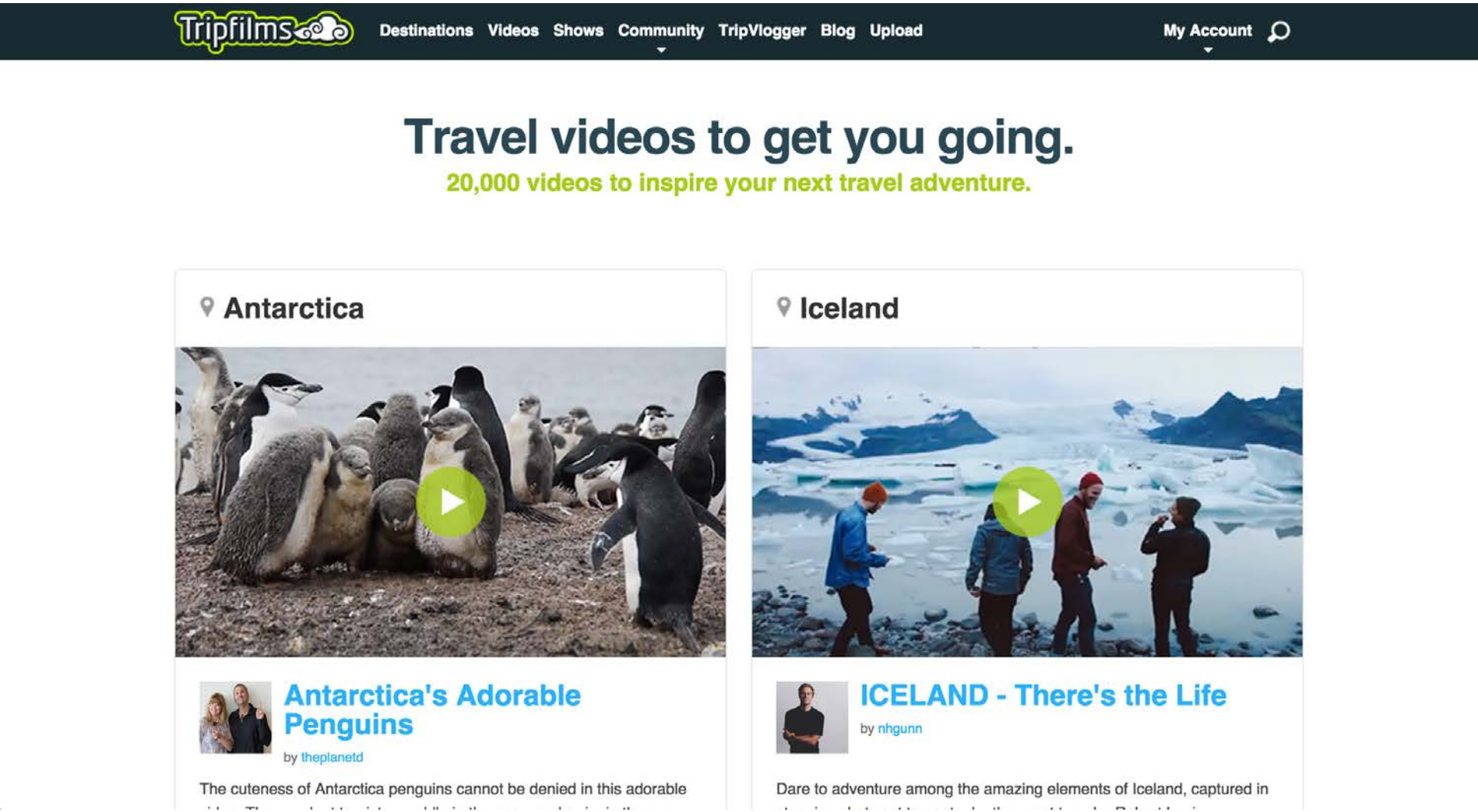
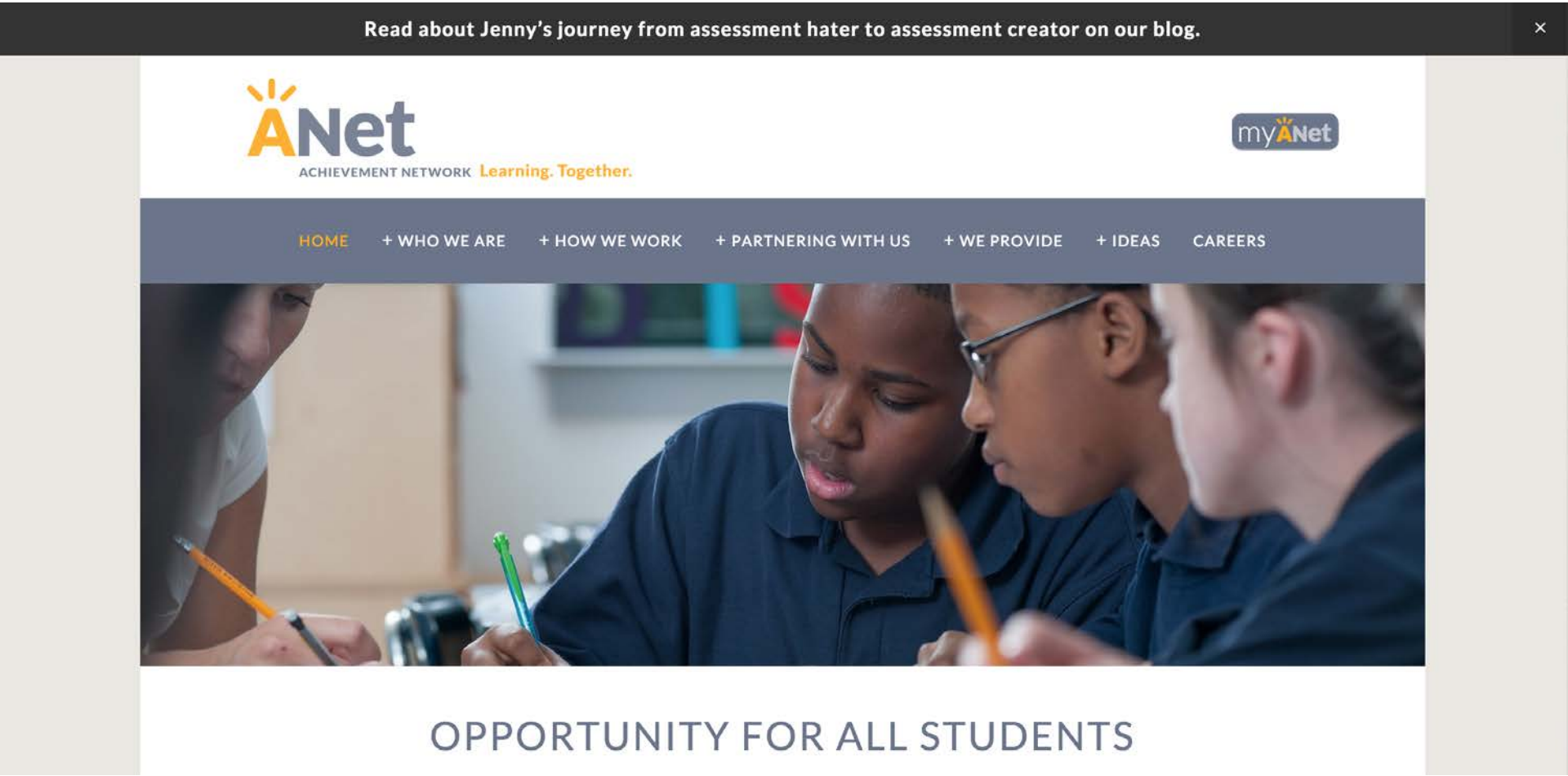
Making Sense of Medicare Data

From Mining To Analytics

Tripfilms.com



The Achievement Network

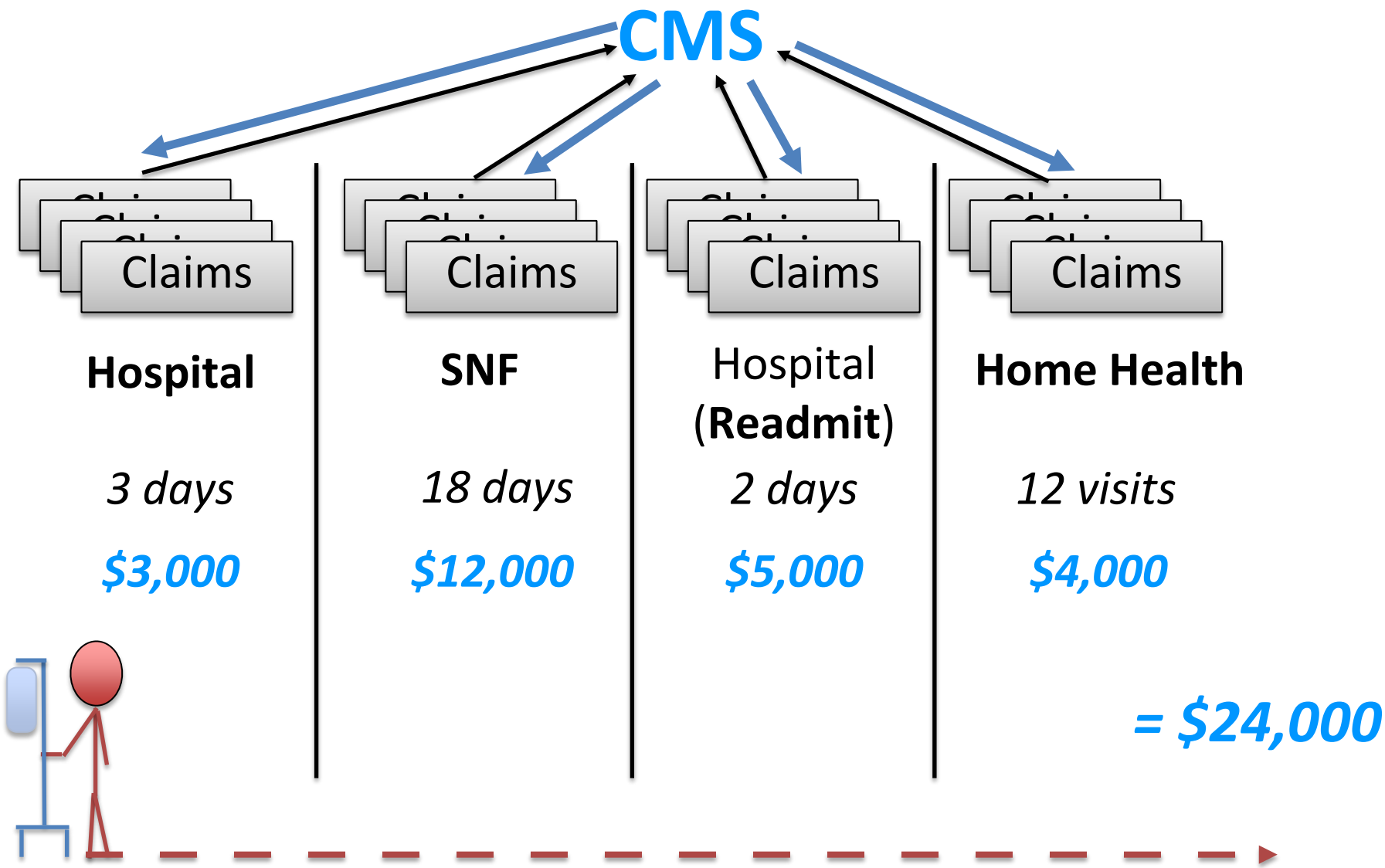


Archway Health Advisors

Medicare

- Centers for Medicare & Medicaid Services (**CMS**)
- **Medicare** is a national social insurance program since 1966 covering Americans aged 65 and older.
- “Fee For Service” Model

Medicare: Fee for Service



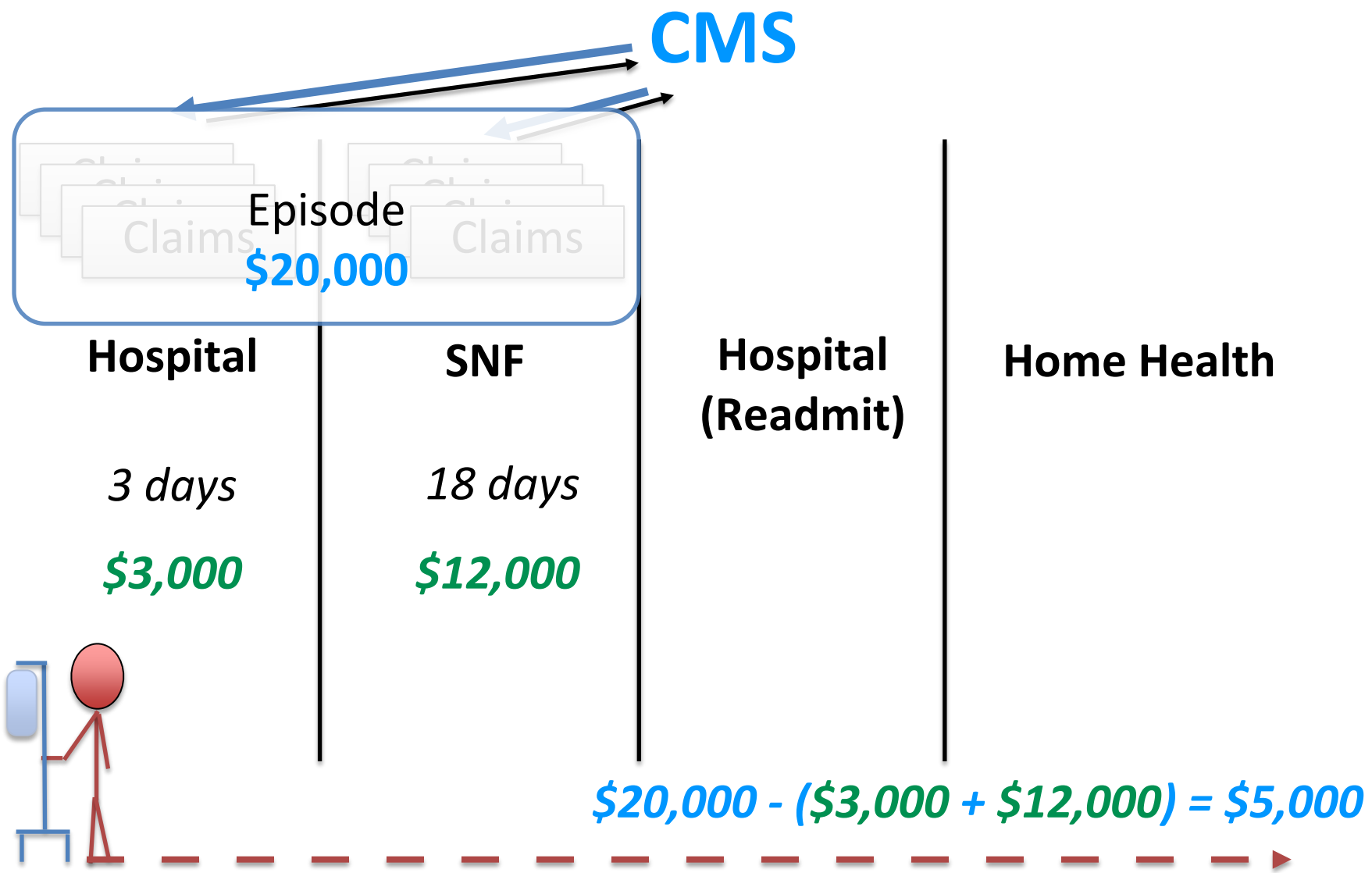
Bundled Payments for Care

- Better care, smarter spending, and healthier people

- The Bundled Payments for Care Improvement (**BPCI**)
  Payment arrangements
  Based on financial and performance

- 4 Models:
  Model 1: Retrospective Acute Care Hospital Stay Only
  Model 2: Retrospective Acute Care Hospital Stay And Post-acute care
  Model 3: Retrospective Post-Acute Care Only
  Model 4: Acute Care Hospital Stay Only

Medicare: BPCI



Claims

- Statement of services and costs from a healthcare provider

  Patient information

  Diagnosis information

  Procedure(s) information

- Types:

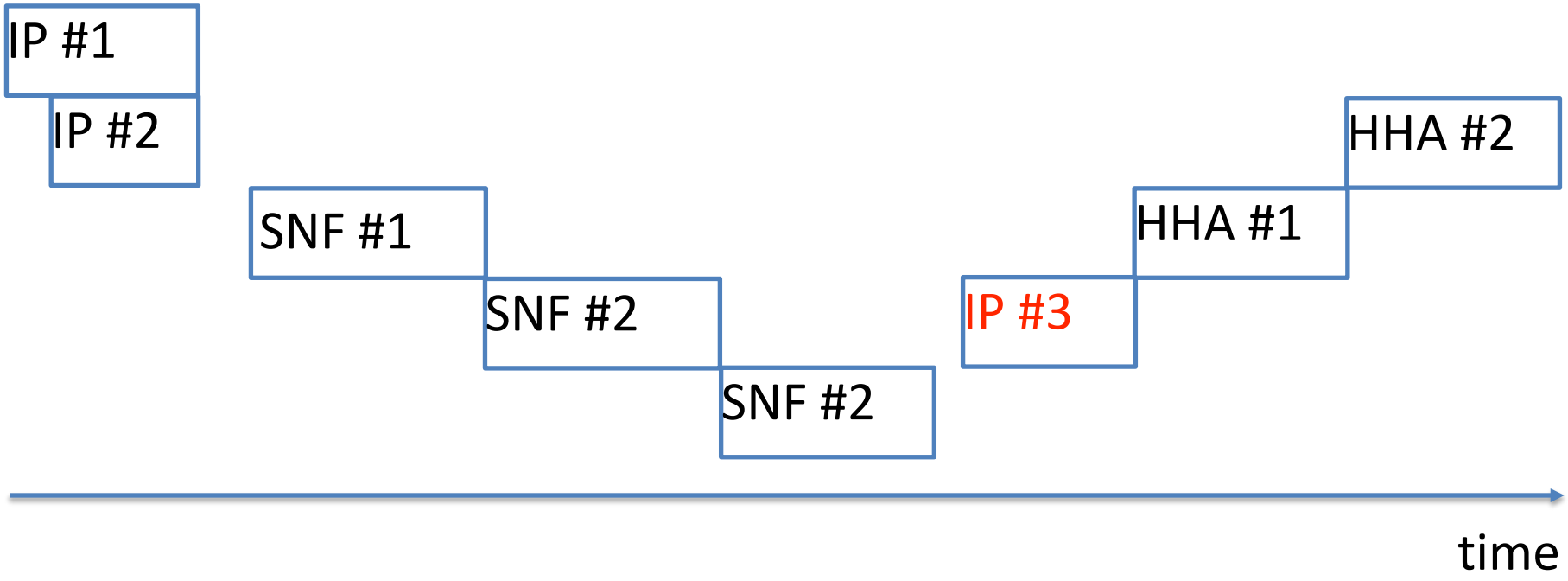
  Hospital (Inpatient) Claims

  Skilled Nursing Facility (SNF) Claims

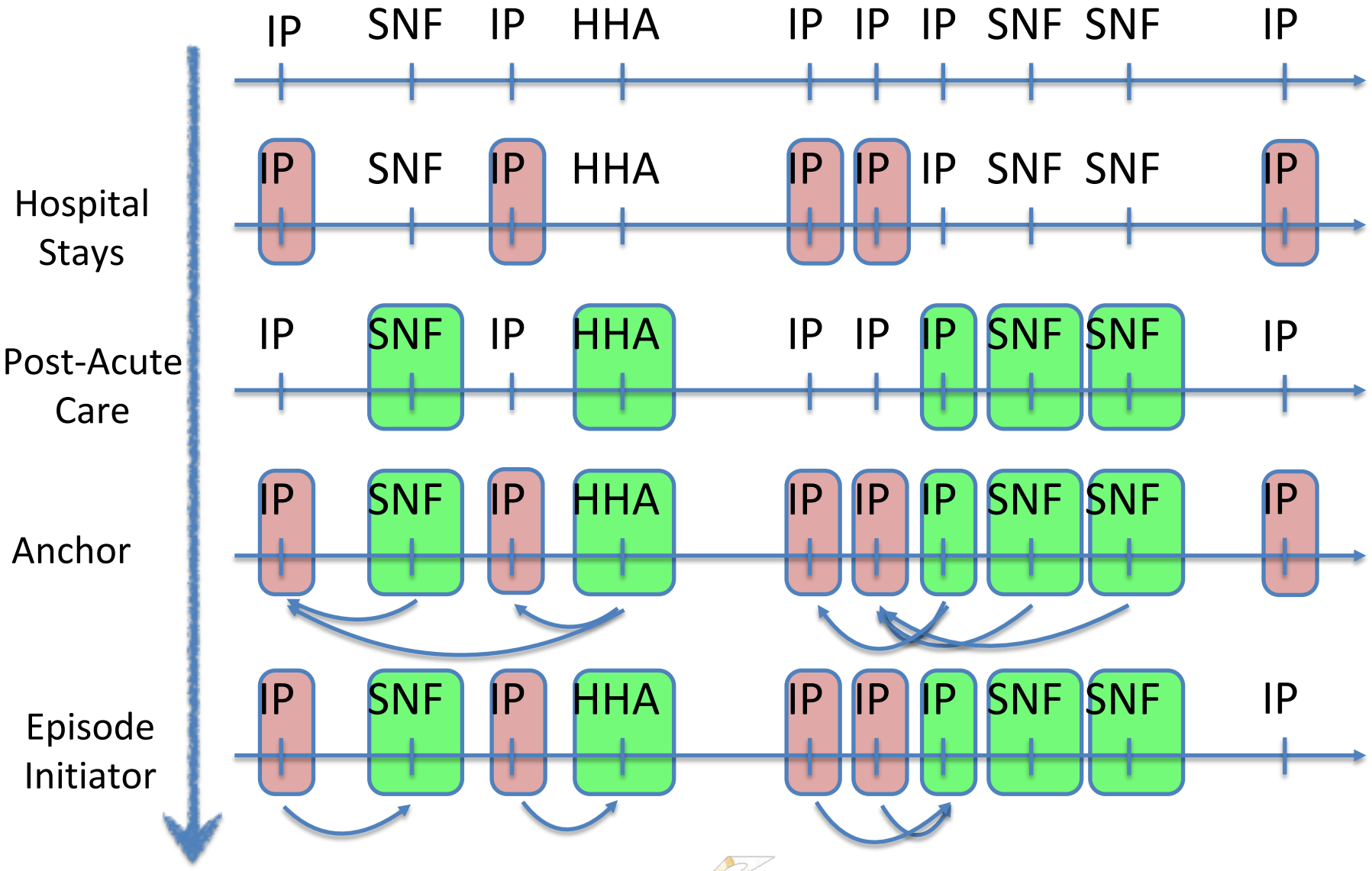
  Home Health Agency (HHA) Claims

  …

Episode of Care



Mining Claims



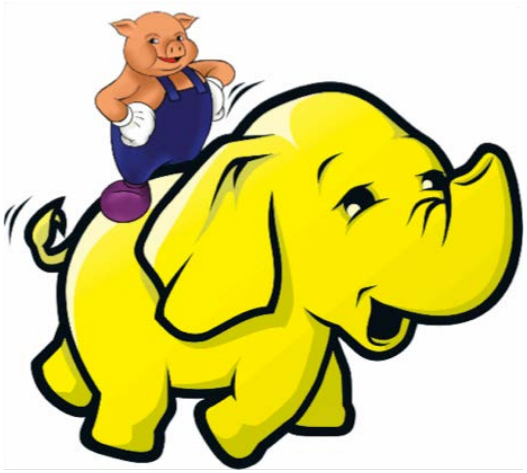
Technologies considered

- **Pig** (& Hadoop)

  Data Processing Language

  Procedural Language

  Relational-oriented



- **SAS**

  Business Analytics & BI Software

  De-facto standard in Healthcare Industry

  Proprietary

sas®

THE POWER TO KNOW.®

ARCHWAY HEALTH ADVISORS

HPCC SYSTEMS®

HPCC Systems

Quick Introduction

Rodrigo Pastrana -Consulting Software Engineer

LexisNexis®

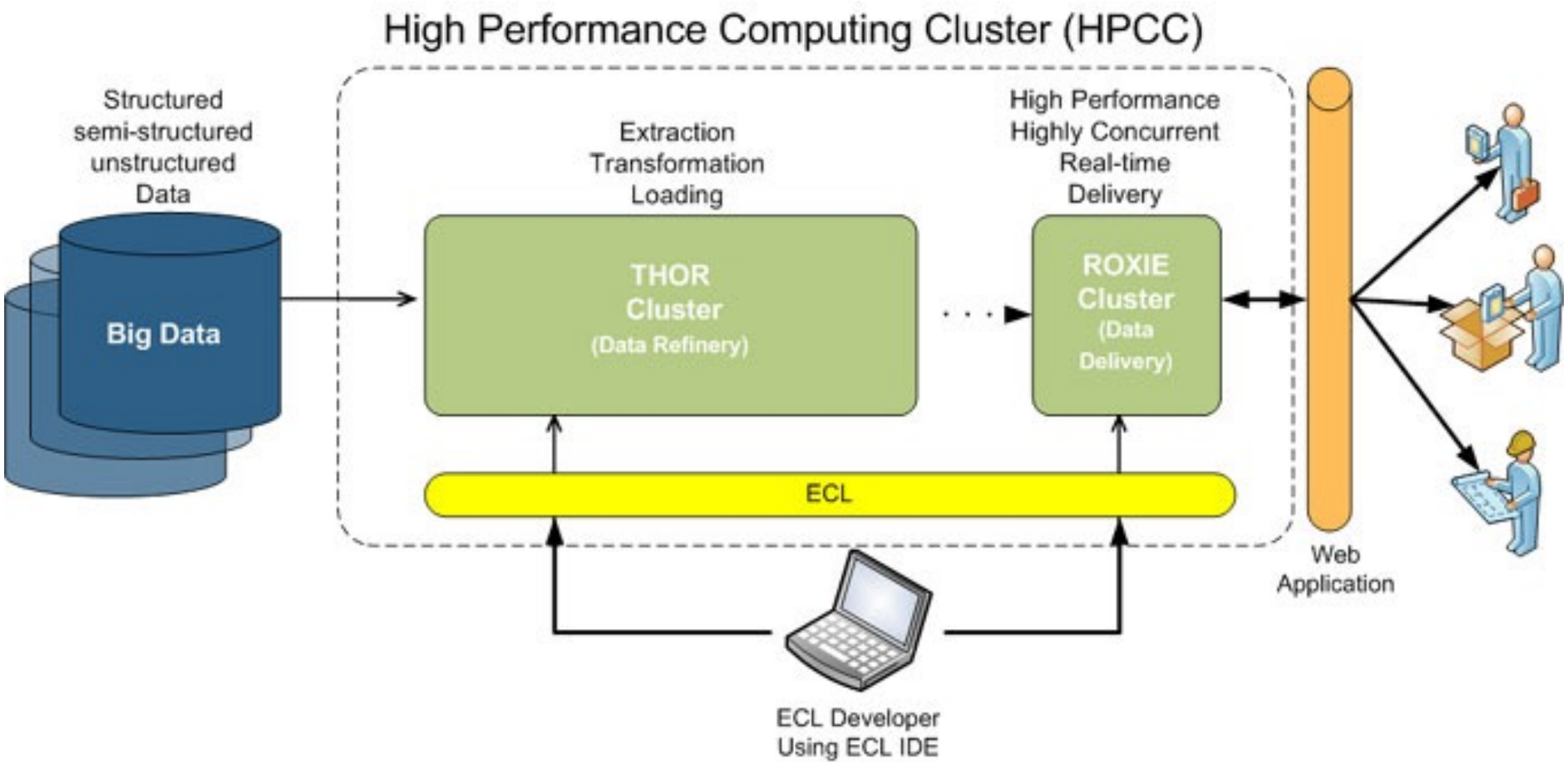
Risk Solutions

What is HPCC Systems

HPCC SYSTEMS®

- Open Source distributed data-intensive computing platform
- Provides end-to-end Big Data workflow management , scheduler, integration tools, etc
- Runs on commodity computing/storage nodes
- Binary packages available for the most common Linux distributions
- Originally designed circa 1999 (predates the original paper on MapReduce from Dec. ‘04)
- Improved over a decade of real-world Big Data analytics
- In use across critical production environments throughout LexisNexis for more than 10 years

The HPCC Systems platform



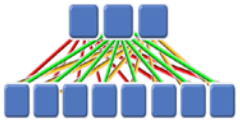
The Three HPCC Systems components

HPCC SYSTEMS®

1 **HPCC Systems Data Refinery (Thor)**

- Massively Parallel data processing engine
- Enables data integration on a scale not previously available
- Programmable using ECL

2 **HPCC Systems Data Delivery Engine (Roxie)**



- A massively parallel, high throughput, query engine
- Low latency, highly concurrent and highly available
- Several advanced strategies for efficient retrieval
- Programmable using ECL

3 **Enterprise Control Language (ECL)**

- An easy to use, declarative data-centric programming language optimized for large-scale data management and query processing
- Highly efficient; automatically distributes workload across all nodes; compiles to native machine code.
- Automatic parallelization and synchronization

Conclusion: End to End platform

- No need for any third party tools

Enterprise Control Language (ECL)

HPCC SYSTEMS®

- **Declarative programming language:** Describe **what** needs to be done and **not how** to do it
- **Powerful:** High level data activities like JOIN, TRANSFORM, PROJECT, SORT, DISTRIBUTE, MAP, etc. are available.
- **Extensible:** Modular and extensible, it can shape itself to adapt to the type of problem at hand
- **Implicitly parallel:** Parallelism is built into the underlying platform. The programmer needs not be concerned with data partitioning and parallelism
- **Maintainable:** High level programming language, without side effects and with efficient encapsulation; programs are more succinct, reliable and easier to troubleshoot
- **Complete:** ECL provides a complete data programming paradigm
- **Homogeneous:** One language to express data algorithms across the entire HPCC Systems platform: data integration, analytics and high speed delivery
- **Polyglottic:** ECL supports the embedding of other languages such as Java, Python, R, SQL, and more

```
// Initialize output log
log_out_init := project(log_init,
                        transform(layout_logo
                        self := left,
                  self := []));

// Create error log
outerrorfile := join(log_seq,
                     log_out_init,
            left.linenum = right.linenum,
            transform(recordof(log_seq),
                      self := left),
            left only,
            hash);

// Denormalize key value pairs
outlogfile := sort(denormalize(distribute(log_
                               sort(distribute(key
                left.linenum = right.linenum,
                transform(layout_logout,
                          self.keyvals := left
                                          row({ric
                     self := left),
```

Current Status and Resources

HPCC SYSTEMS®

- HPCCSystems.com – Tutorials, Docs, Platform distributions, and more
- Latest release 5.2.0 adds many new features and improvements
  - Drastic GUI improvements
  - Ganglia and Nagios plug-in for system monitoring and alerting
  - Security Enhancements – tighter authentication measures, intra-component communication encryption
  - Embedded Languages – Cassandra support, memcache and redis access
  - JSON based data support
  - Dynamic ESDL – Provides simple middleware/back-end interface definition
  - JAVA API project – facilitates interaction between Java based apps and HPCC web services and c++ tools
- Available now – HPCCSystems.com

Data mining with HPCC Systems

- Thor

  Responsible for processing vast amount of data

  Optimized for Extraction, Transformation, Loading, Sorting and Linking Data

- ECL

  Declarative

  More Data Centric

  Fast & Implicitly Parallel

  Inline data

  Unit Tests in ECL

ARCHWAY HEALTH ADVISORS

SQL vs ECL

SQL

```
SELECT
    diag_group_cd,
    COUNT(*) as volume
    SUM(pmt_amt) as costs
FROM
    inpatient_claims
GROUP BY
    diag_group_cd;
```

ECL

```
TABLE(
    inpatient_claims,
     {
         diag_group_cd;
         INTEGER volume :=
COUNT(GROUP);
         REAL costs := SUM(pmt_amt);
    },
    diag_group_cd
);
```

```
SELECT
     *
FROM
    inpatient_claims LEFT
    JOIN ip_value_codes
RIGHT
        ON LEFT.id = RIGHT.id
```

```
JOIN(
    inpatient_claims,
    ip_value_codes,
    LEFT.id = RIGHT.id
);
```

ARCHWAY HEALTH ADVISORS

SQL vs ECL

SQL

```
DECLARE my_cursor CURSOR FOR
SELECT * FROM inpatient_claims;

OPEN my_cursor

FETCH NEXT FROM my_cursor
INTO @..., @...

WHILE @@FETCH_STATUS = 0
BEGIN
      ...
END
CLOSE my_cursor;
DEALLOCATE my_cursor;
```

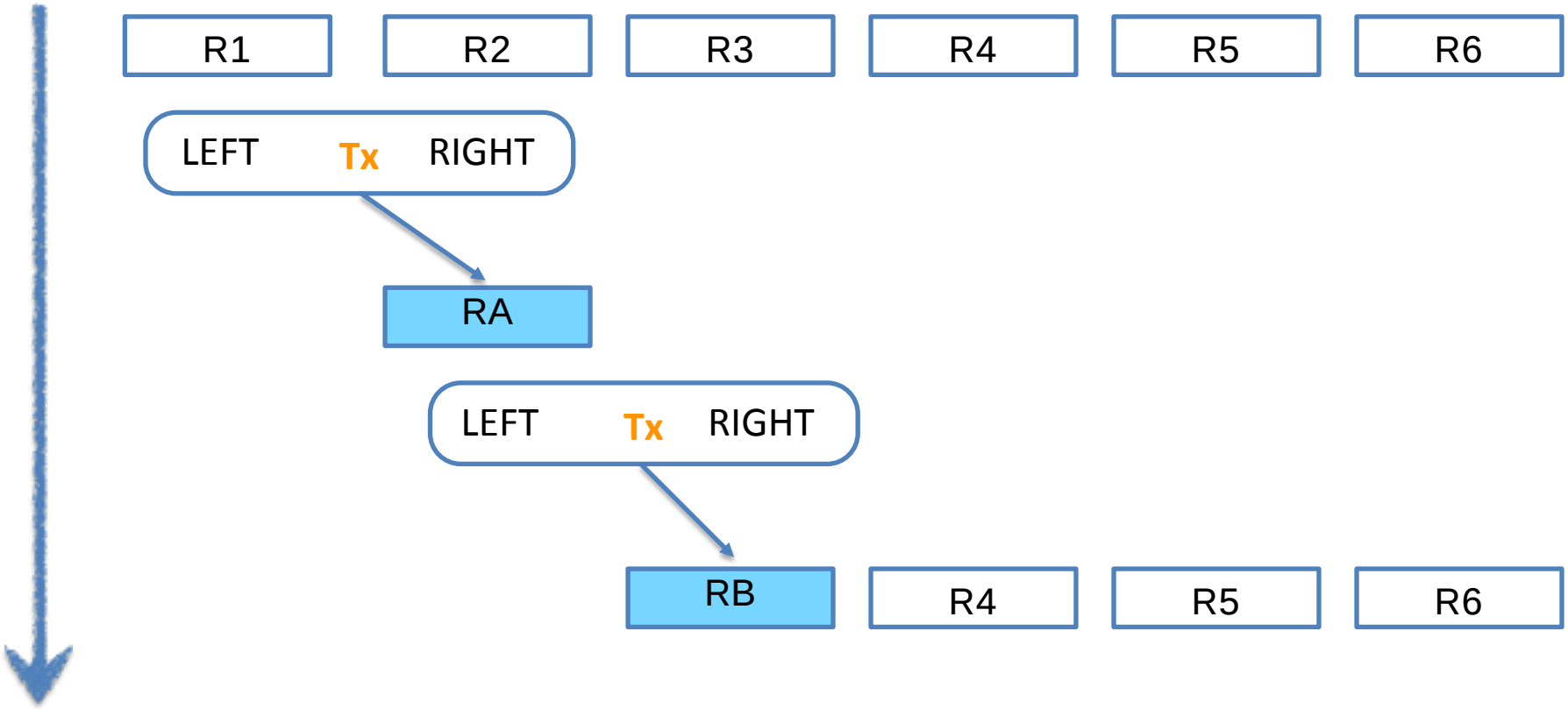
ECL

```
ITERATE(
    inpatient_claims,
    TRANSFORM(inpatient_claim_layout,
        SELF.is_dropped :=
is_one_year_or_greater(
        RIGHT.admsn_dt,
        RIGHT.dschrgdt);

        SELF := RIGHT;
    )
);
```

ARCHWAY HEALTH ADVISORS

ECL ROLLUP

ROLLUP(dataset, condition(**LEFT**, **RIGHT**), transformation(**LEFT**, **RIGHT**))



Processing Claims

1. The intent here is to make the series of interim claims look like a single claim for most purposes, where the admission date of the first claim becomes the admission date of the whole claim and the discharge date of the last claim in the series becomes the discharge date of the whole claim.
2. The admission date from the first series in the claim and the discharge date from the last series in the claim define the length of the stay.
3. The MS-DRG from the last claim in the single stay (the discharge MS-DRG) determines whether the hospital stay becomes an anchor record, or whether the stay is included/excluded as a readmission for an existing episode.
4. Costs across all IP claims included in the single stay are aggregated to the stay level.
5. Claims where the last in the series of claims has patient (…) [as “still a patient”, not discharged], flag these and drop all of the claims in the series from the IP hospital stay file.

Processing Claims With ECL

```
H_1 := SORT( A , bene_sk, provider, admsn_dt, dschrgdt, thru_dt);
H_2 := ROLLUP(H_1,
   is_interim(LEFT, RIGHT),
   merge_interim_claims(LEFT, RIGHT));

H_3 := JOIN(H_2, H_1, LEFT.bene_sk = RIGHT.bene_sk [...], RIGHT ONLY);
H_4 := PROJECT(H_3, TRANSFORM(BPCI.Layouts.ip_claim_etl_layout,
          SELF.is_dropped := TRUE;
          SELF.dropped_reason_code := BPCI.Layouts.DROPPED_REASON_CODES.InterimClaim;
          SELF := LEFT;
));

H := H_2 + H_4;
```

Template Language

```
EXPORT load_all_client_files(pId, pFileSet, pBaseDataDirectory) := MACRO
    LOADXML(pFileSet);
    baseDataDirectory := pBaseDataDirectory + pId + '/';
    #FOR(folder)
        #UNIQUENAME(subId)
        %subId% := %''%;
        #UNIQUENAME(subDS)
        %subDS% := Client.Datasets(%subId%);

        [...]

        #UNIQUENAME(id)
        %id% := pId + '::' + %''%;
        #UNIQUENAME(dataDir)
        %dataDir% := %baseDataDirectory% + %''% + '/';
        #UNIQUENAME(etl)
        %etl% := Client.ETL(%dataDir%, %id%);
        %etl%.run();
    #END
ENDMACRO;
```

Template Language

```
file_set := '<folders>' +
    '<folder>M201409</folder>' +
    '<folder>M201410</folder>' +
    '<folder>M201411</folder>' +
    '<folder>M201412</folder>' +
    '<folder>M201501</folder>' +
    '<folder>M201502</folder>' +
    '<folder>M201503</folder>' +
'</folders>';

load_all_client_files(1234, file_set, '/volume1/data/');
```

Beyond Processing Data

- Security & Authentication
- Collaboration
- Unit Tests
- Visualizations

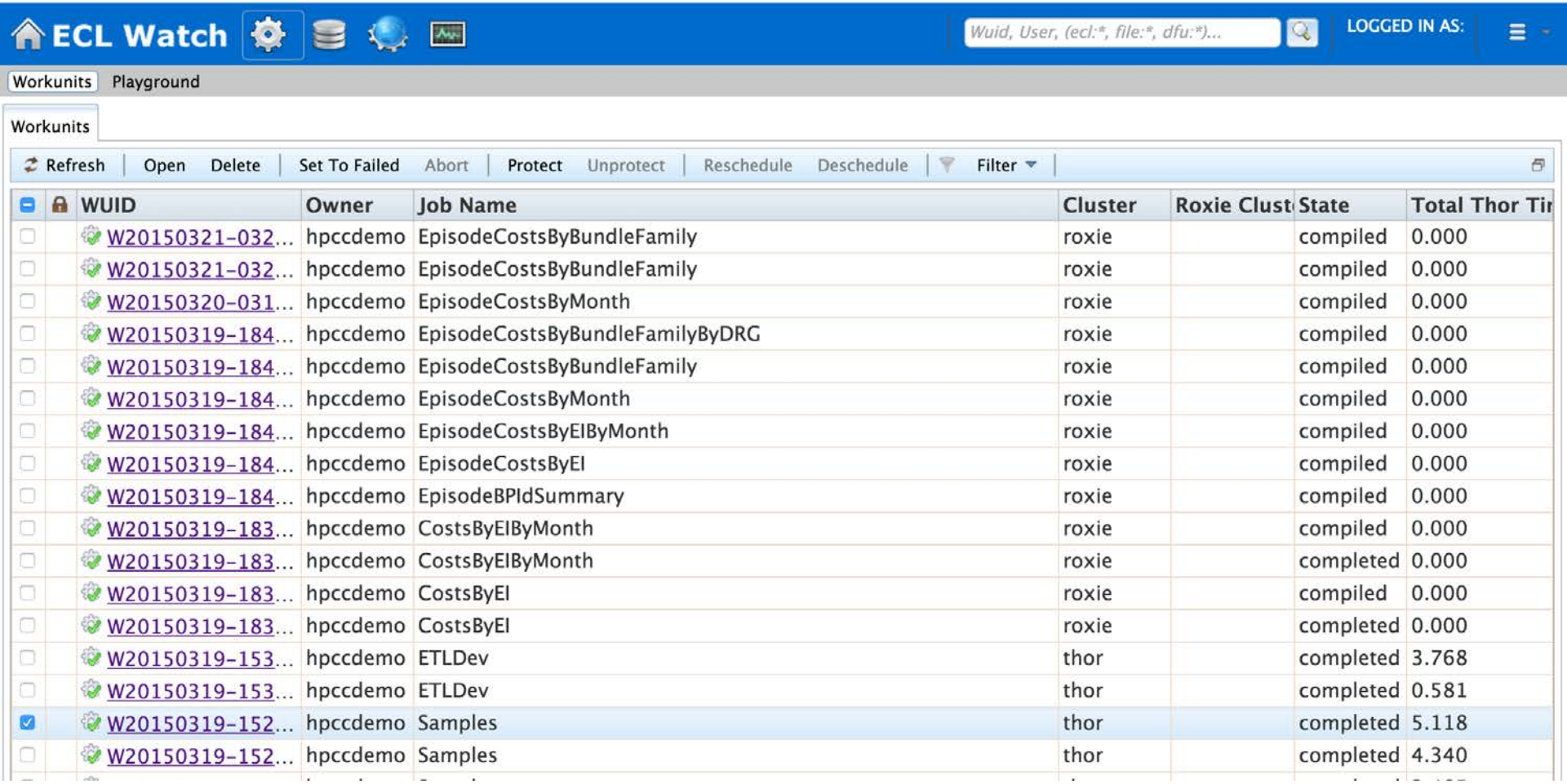
Beyond Processing: Security

- HTTPS
- Htpasswd
- LDAP support
- File level security when using LDAP

Beyond Processing: Workunits

- Workunit Identifier
- Attribution
- Query
- Timings
- Results

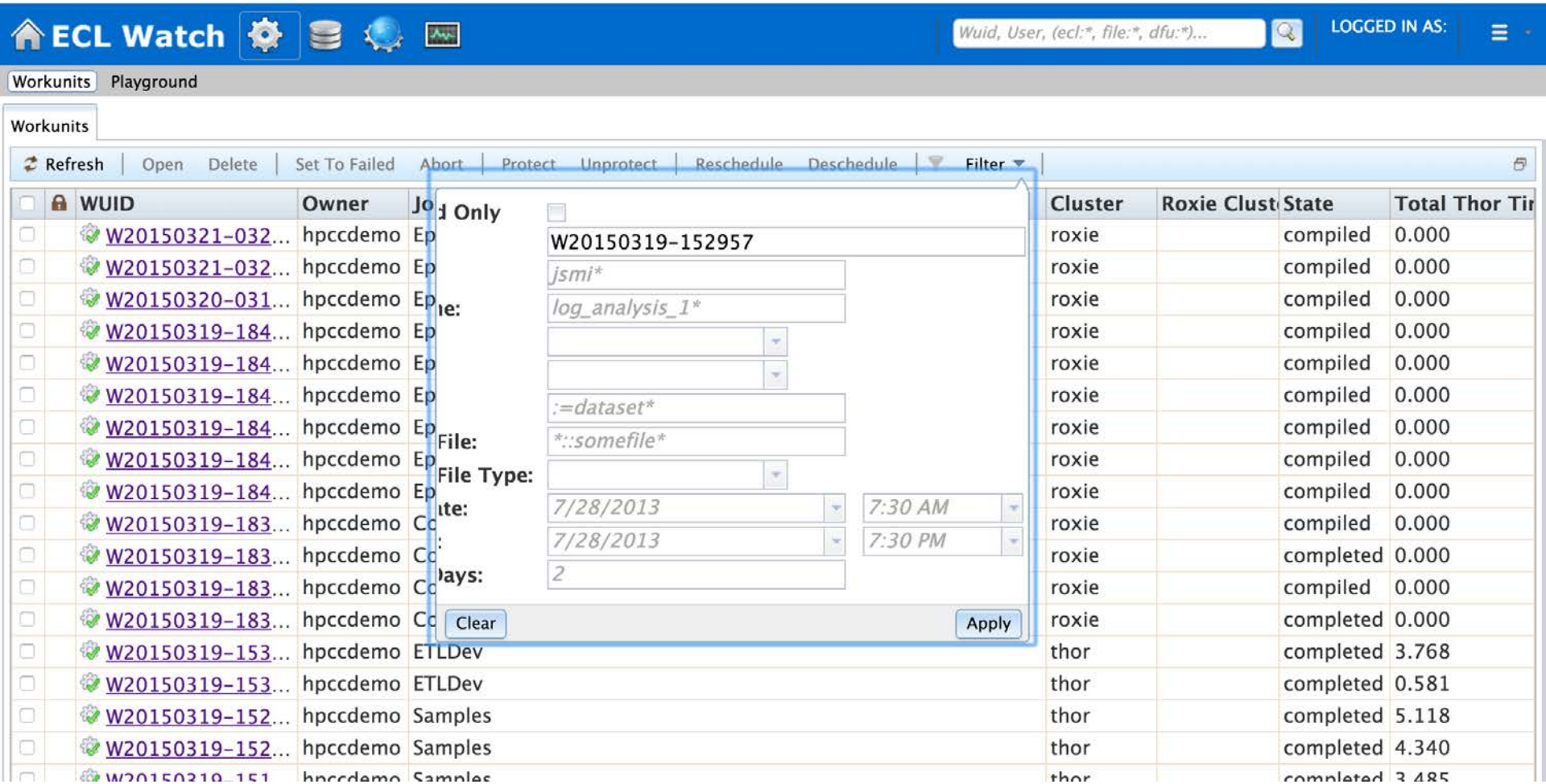
Beyond Processing: Collaboration



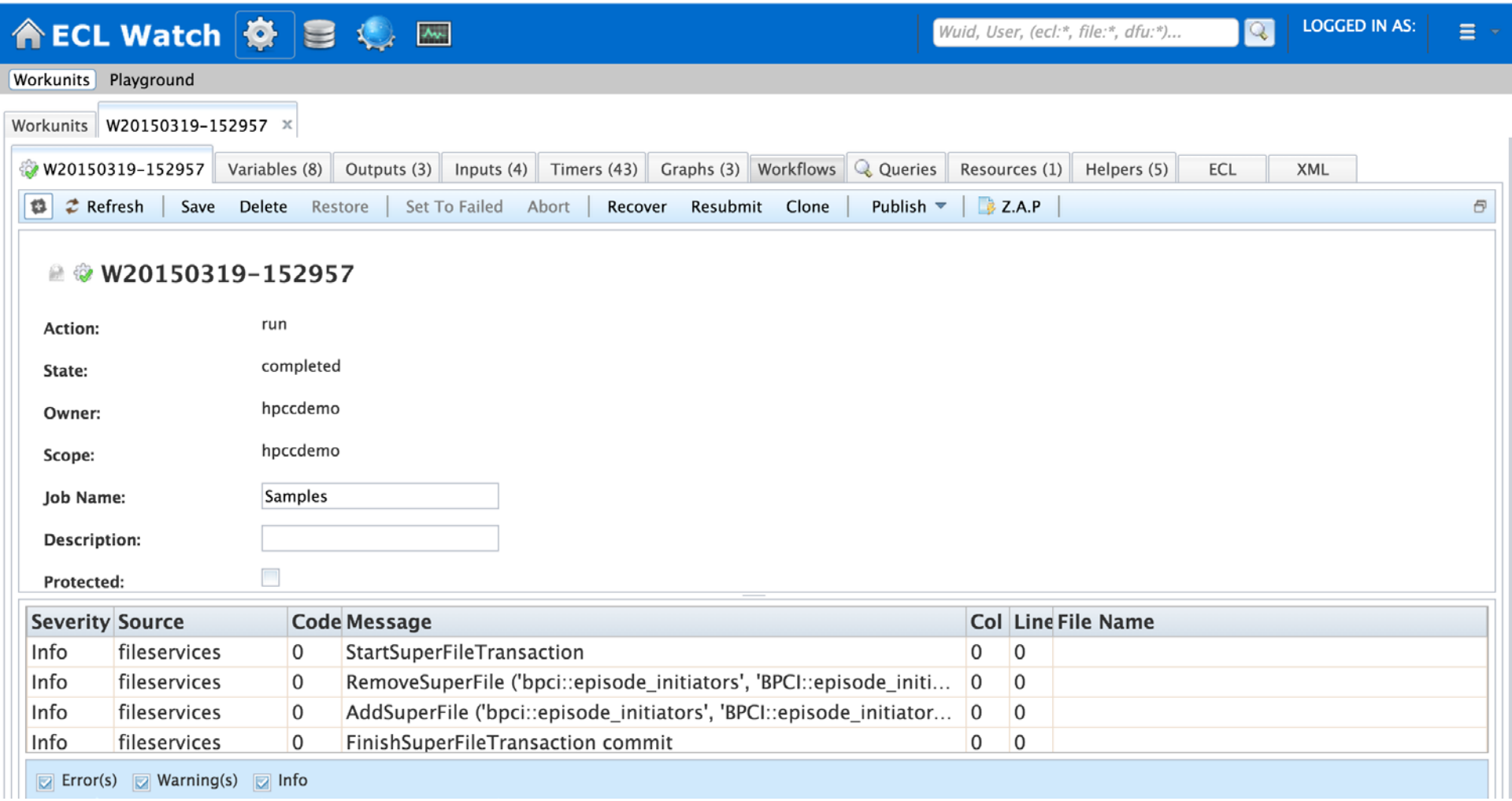
WUID	Owner	Job Name	Cluster	Roxie Clust	State	Total Thor Tir
W20150321-032...	hpccdemo	EpisodeCostsByBundleFamily	roxie		compiled	0.000
W20150321-032...	hpccdemo	EpisodeCostsByBundleFamily	roxie		compiled	0.000
W20150320-031...	hpccdemo	EpisodeCostsByMonth	roxie		compiled	0.000
W20150319-184...	hpccdemo	EpisodeCostsByBundleFamilyByDRG	roxie		compiled	0.000
W20150319-184...	hpccdemo	EpisodeCostsByBundleFamily	roxie		compiled	0.000
W20150319-184...	hpccdemo	EpisodeCostsByMonth	roxie		compiled	0.000
W20150319-184...	hpccdemo	EpisodeCostsByEIByMonth	roxie		compiled	0.000
W20150319-184...	hpccdemo	EpisodeCostsByEI	roxie		compiled	0.000
W20150319-184...	hpccdemo	EpisodeBPIdSummary	roxie		compiled	0.000
W20150319-183...	hpccdemo	CostsByEIByMonth	roxie		compiled	0.000
W20150319-183...	hpccdemo	CostsByEIByMonth	roxie		completed	0.000
W20150319-183...	hpccdemo	CostsByEI	roxie		compiled	0.000
W20150319-183...	hpccdemo	CostsByEI	roxie		completed	0.000
W20150319-153...	hpccdemo	ETLDev	thor		completed	3.768
W20150319-153...	hpccdemo	ETLDev	thor		completed	0.581
W20150319-152...	hpccdemo	Samples	thor		completed	5.118
W20150319-152...	hpccdemo	Samples	thor		completed	4.340

ARCHWAY HEALTH ADVISORS

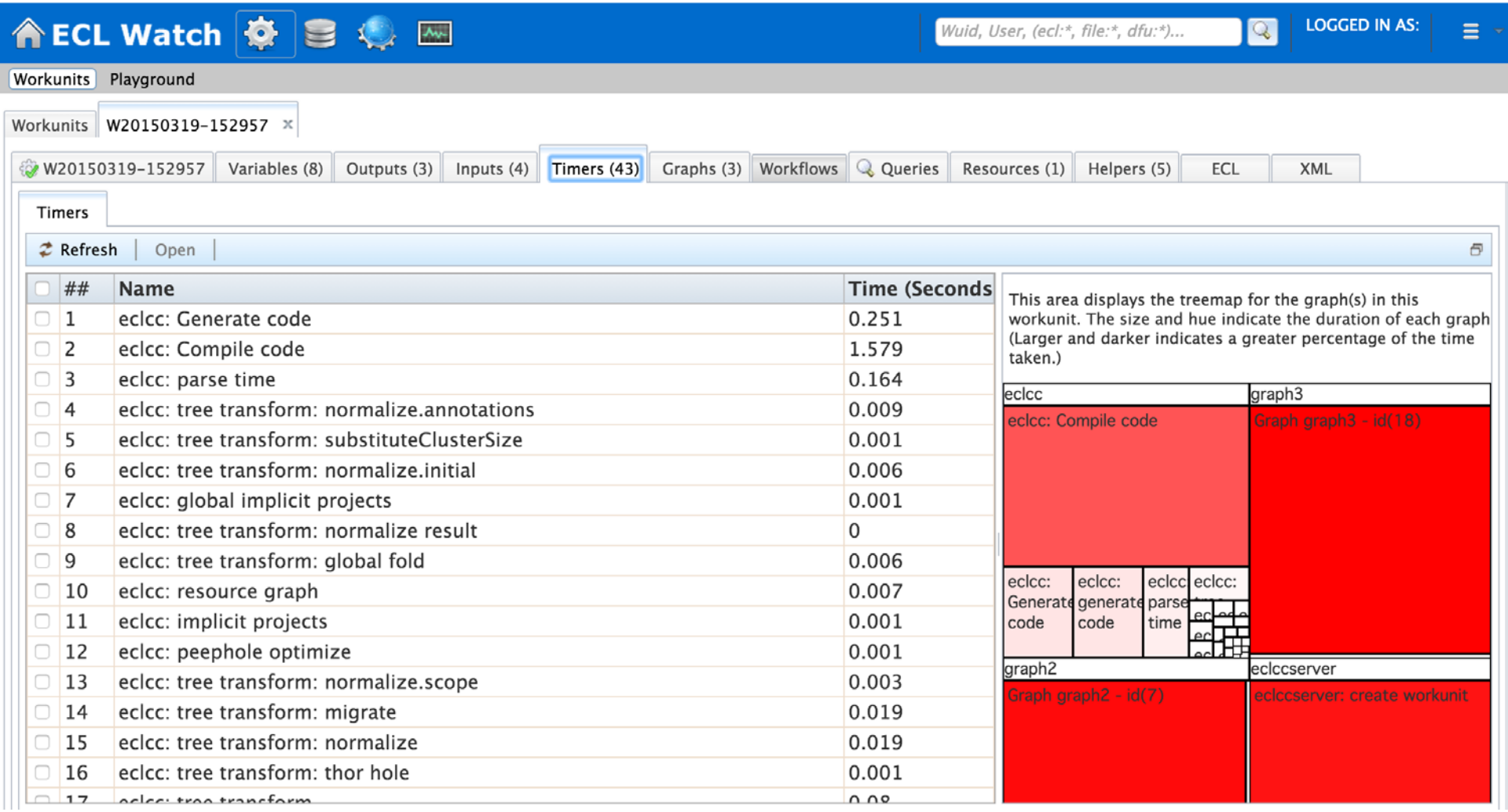
Beyond Processing: Collaboration (2)



Beyond Processing: Collaboration



Beyond Processing: Collaboration



Beyond Processing: Unit Tests

```
interim_claims := MODULE

    // Test Data
    test_set :=
        BPCI.Test.Samples.ip_claim(
            bene_id := 1, claim_id := 1, pmt_amt  := 3042.0, ...)
        + BPCI.Test.Samples.ip_claim(
            bene_id := 1, claim_id := 2, pmt_amt  := 11409.0, ...)
        + ....
    ;
    ...
    EXPORT Actual := Step2.ip_stays;

    SHARED TestSuite := MODULE
        EXPORT Test01 :=  ASSERT(oActual(NOT is_dropped), claimno IN [1,2], 'Did not filter o
        EXPORT Test02 :=  ASSERT(oActual(is_dropped), claimno IN [3,5]);
    END;

    EXPORT AllTests := TestSuite.Test01 + TestSuite.Test02;

END;
```

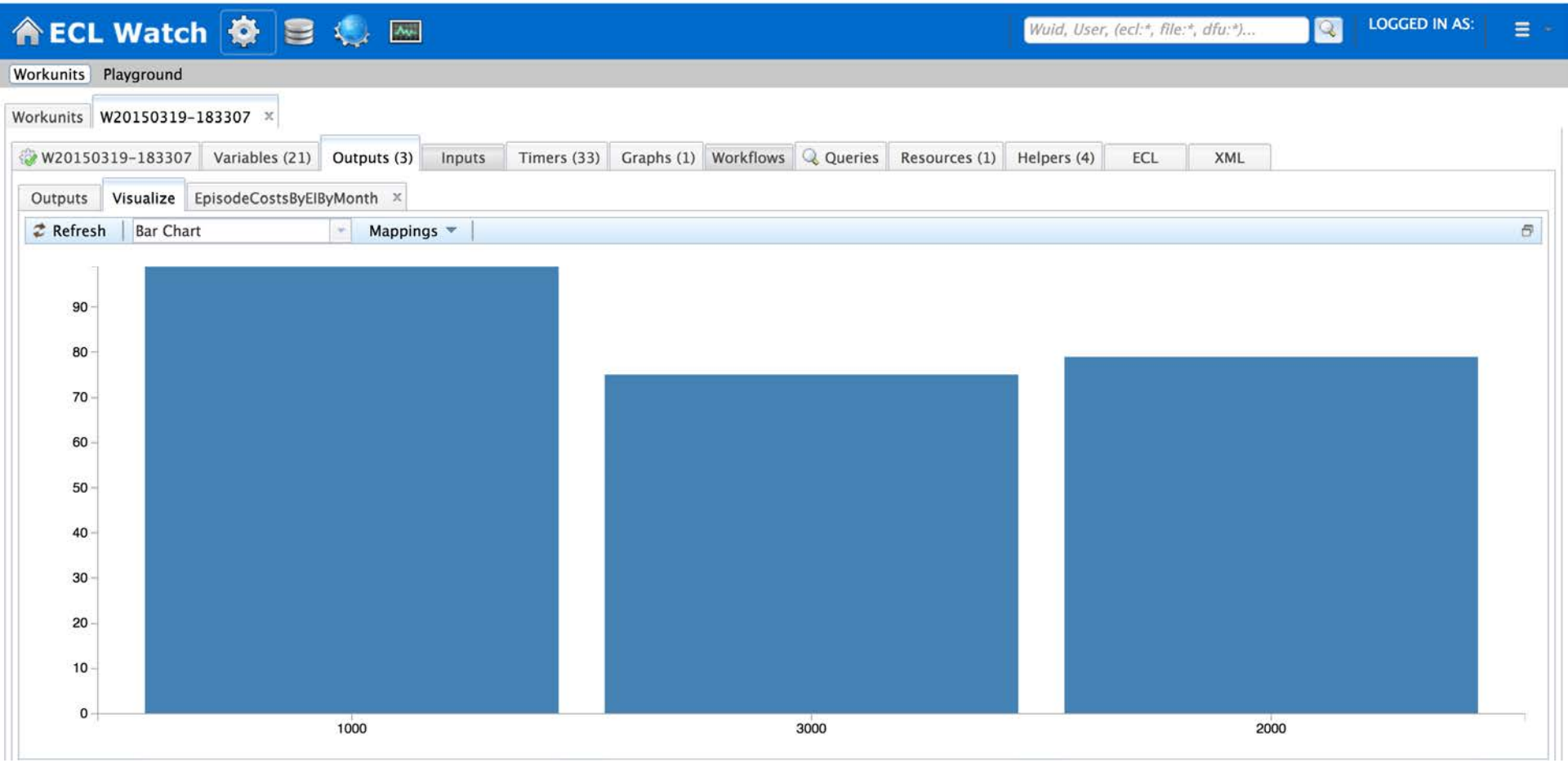
Beyond Processing: Unit Tests (2)

```
simplified_ip_layout := RECORD
  UNSIGNED bene_id;
  UNSIGNED claim_id;
  STRING claim_type;
  STRING provider_number;
  INTEGER4 through_date;
  STRING status_code;
  INTEGER4 admission_date;
  INTEGER4 discharge_date;
  STRING ms_drg_code;
END;
```

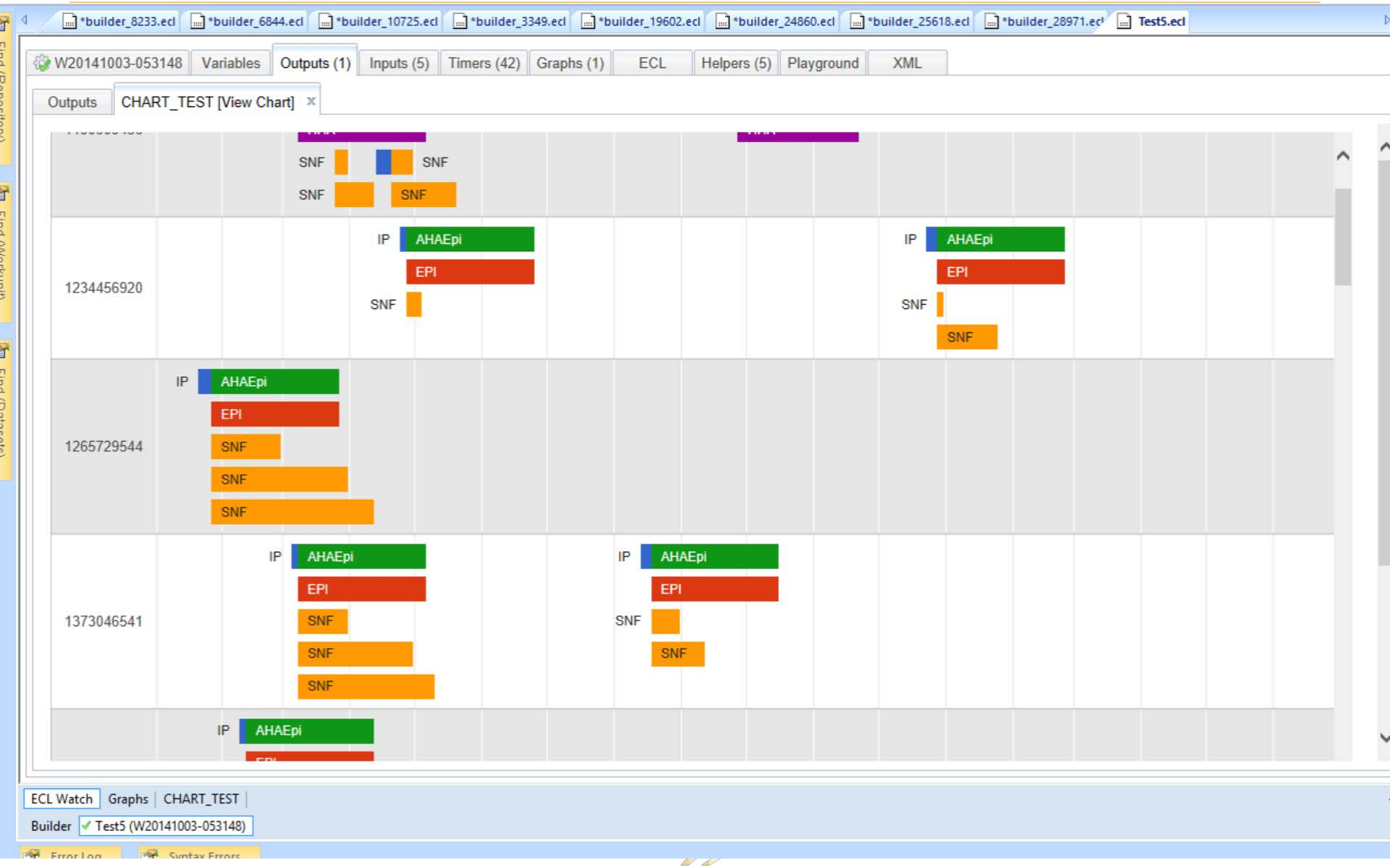
```
// Using inline dataset
simple_ip_claims := DATASET([

{1,1,'0','010001',20000201,'',20000120,200
00201,'61'},
  ], simplified_ip_layout);


ip_claims :=
Samples.ip_claims(simple_ip_claims);



// OR passing NAMED parameters
ip_claims := Samples.ip_claims2(
      bene_id := 1,
      claim_id := 1,
      claim_type := '00'
      )
```

ARCHWAY HEALTH ADVISORS

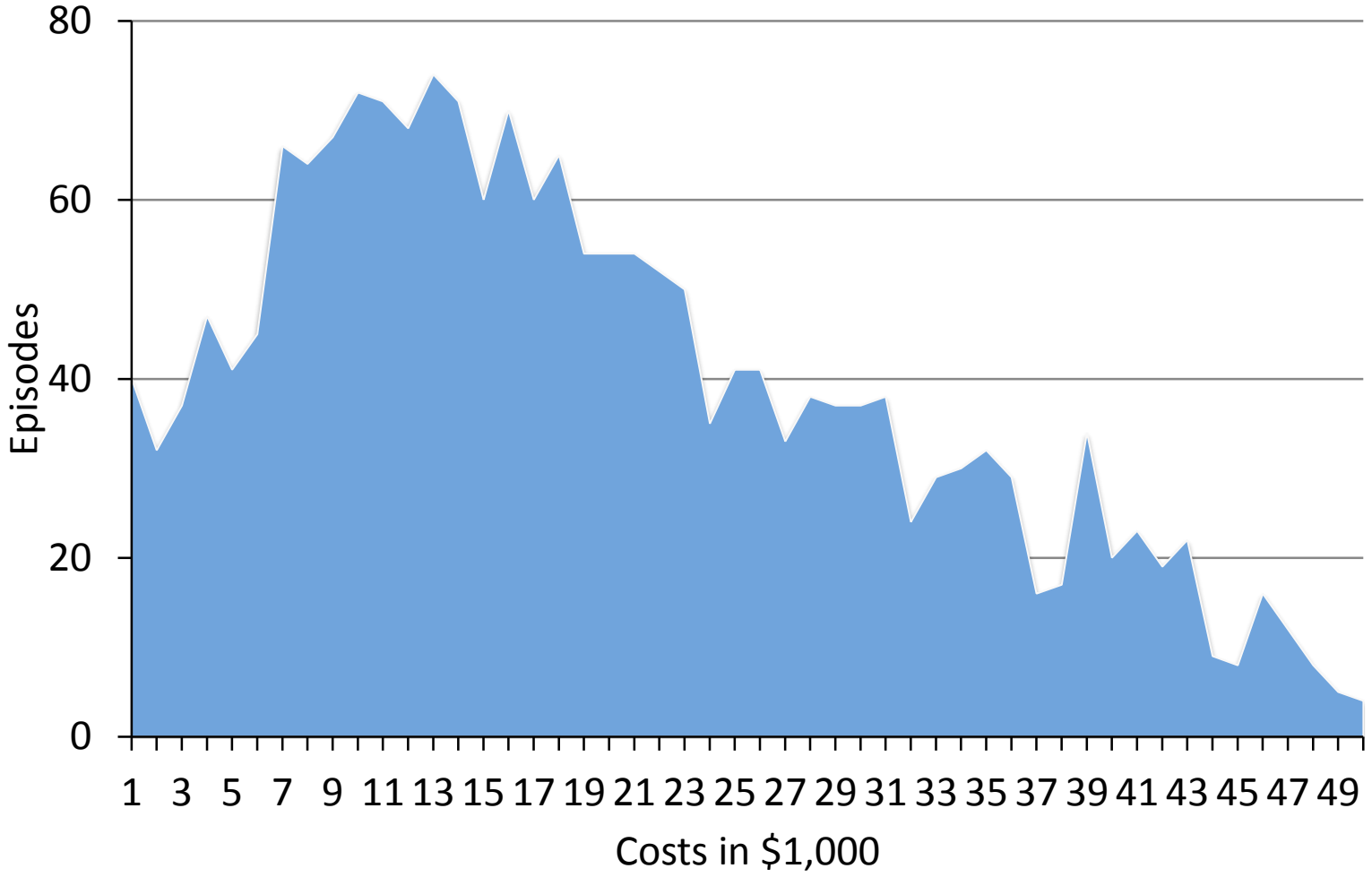
Beyond Processing: Visualization



Custom Visualization

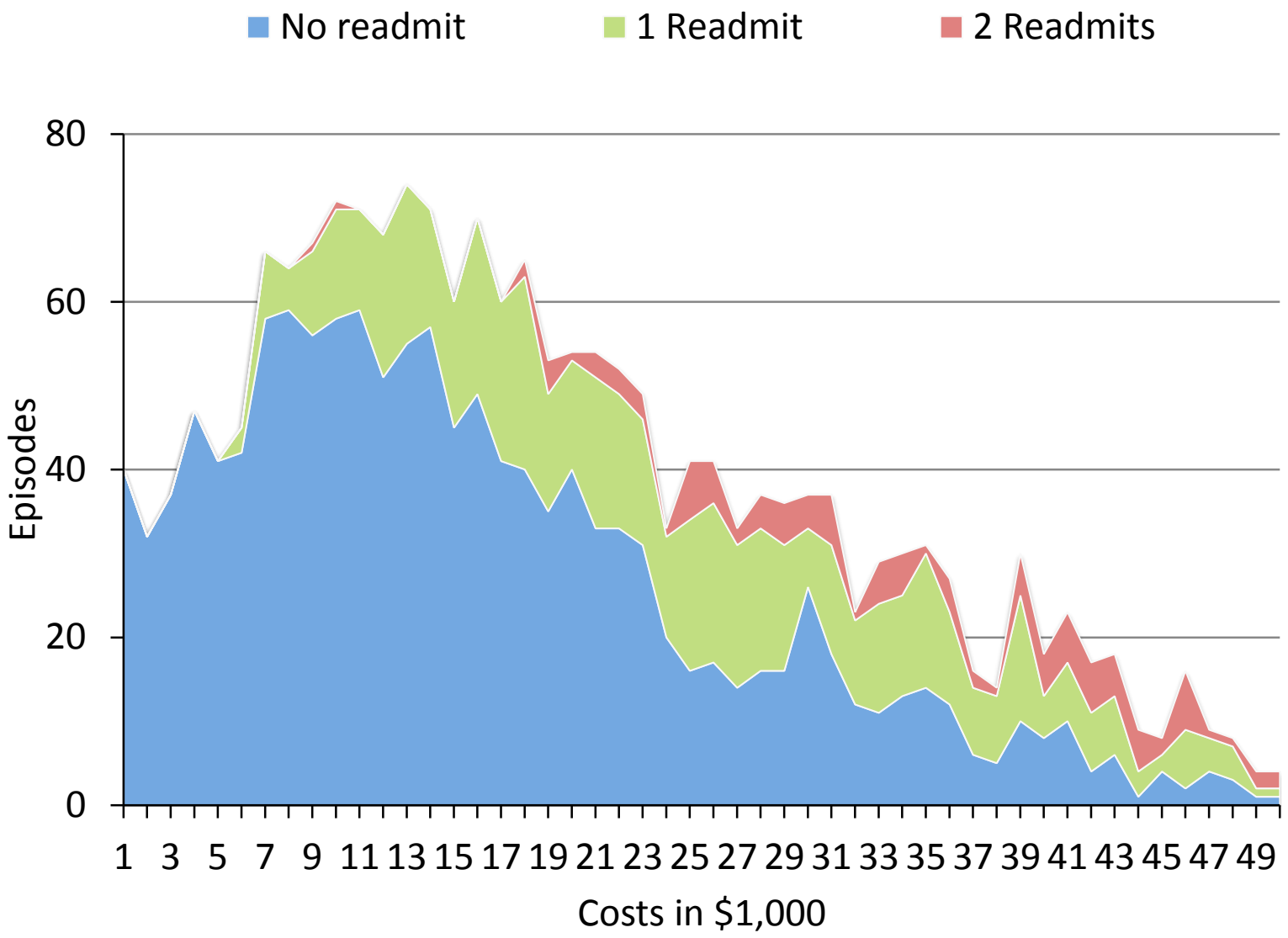


(No) Insights



ARCHWAY HEALTH ADVISORS

Insights



ARCHWAY HEALTH ADVISORS

Data Delivery: Roxie

- Data Delivery Engine
- Indexed, compressed and in-memory
- Data Warehouse Capabilities
- Data Services

Data Services

- Web Services over Data Warehouse
- XML/SOAP but also JSON
- Web Services defined in ECL
- Solution to add/remove data from the cluster

Data Service: Query

- Define service using ECL

```
INTEGER4 oOffset := 1 : STORED('Offset');
INTEGER4 oResults := 100 : STORED('Results');
INTEGER4 oStartDate := 20130201 : STORED('Begin_Date');
INTEGER4 oEndDate := 20140201 : STORED('End_Date');
...

oParams := DATASET([{ oOffset, oResults, oStartDate, oEndDate, ... }],
Layouts.service_parameters_layout);

T(DATASET(RECORDOF(Datasets.dsFactEpisodeCostsIndex)) pData) := FUNCTION
      RETURN TABLE(pData,
            {
                  STRING bpid := pData.bpid;
                  UNSIGNED INTEGER1 model := pData.model;
                  UNSIGNED INTEGER1 post_dsch_prd_length := pData.post_dsch_prd_length;

                  INTEGER8 total_episodes := COUNT(GROUP);
                  DECIMAL15_2 total_costs := SUM(GROUP, sum_post_dsch_prd_pay);
                  DECIMAL15_2 average_costs := AVE(GROUP, sum_post_dsch_prd_pay);
                  DECIMAL15_2 std_dev_costs := SQRT(VARIANCE(GROUP, sum_post_dsch_prd_pay));
            }, bpid, model, post_dsch_prd_length);
END;

ReportServices.BaseService.run_it('Summary', oParameters, T, bpid);
```

Data (Web) Services

https://.../WsEcl/forms/json/query/roxie/summary

Request

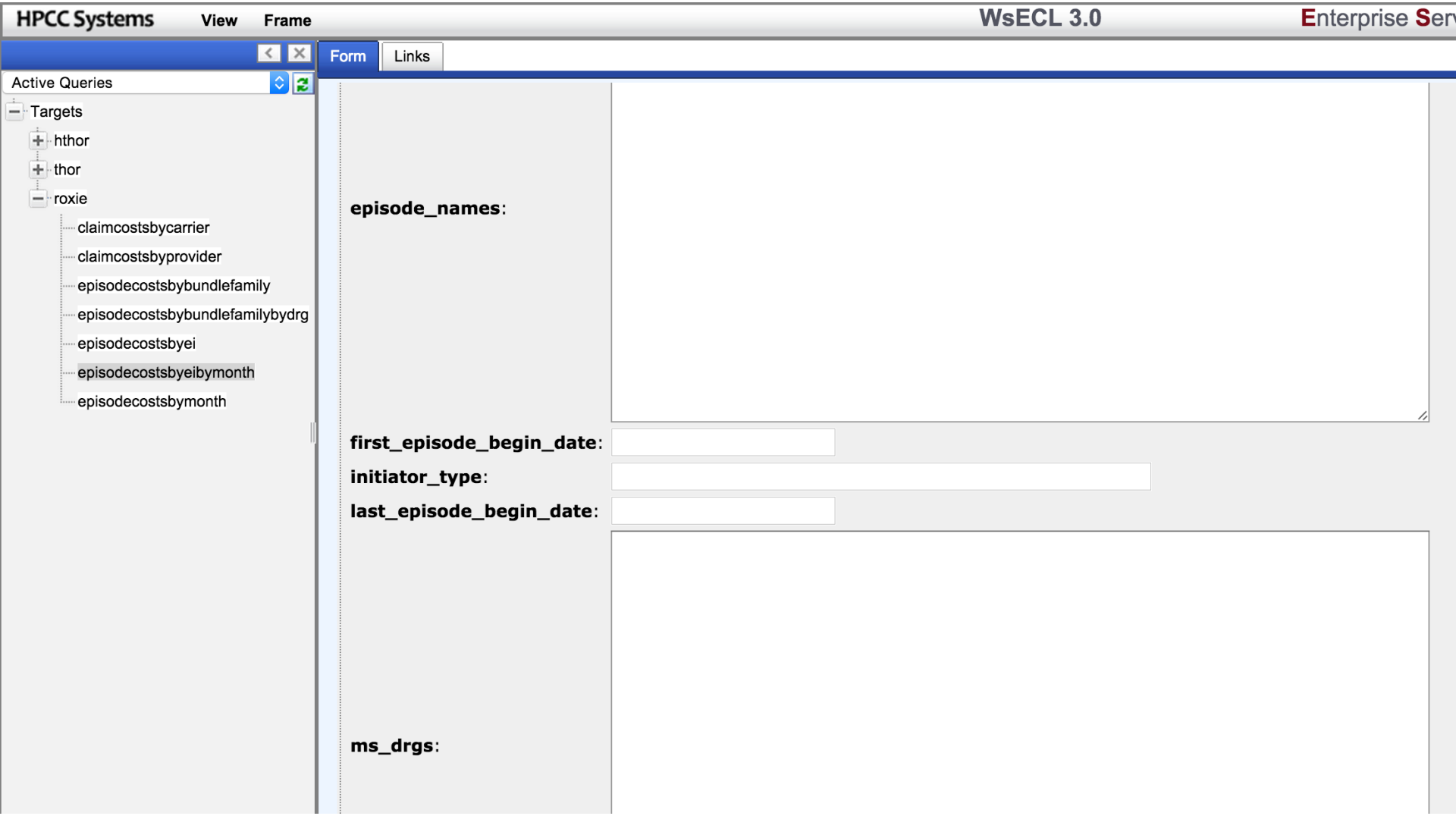
```
{
  "summary": {
    "offset": 1,
    "results": 10,
    "begin_date": 20130101,
    "end_date": 20130201,
    ...
  }
}
```

Response

```
{
 "summaryResponse": {
    ...
    "Results": {
      ...
      "Summary": {
        "Row": [
          {
            "bpid": "9999",
            "model": 2,
            "post_dsch_prd_length": 90,
            "total_episodes": 987,
            "total_costs": 9876543.21,
            "average_costs": 12358.13
            ...
          }
        ]
    ...
```

ARCHWAY HEALTH ADVISORS

Data Services: WsECL



Data Services: WsECL

Loading up data

- Logical vs Physical

  ~abc::subfolder::subsubfolder::myfile

  /abc/subfolder/subsubfolder/myfile

- ECL to load data into cluster:

```
oDS := DATASET(
      std.File.ExternalLogicalFilename('172.0.0.1','/var/lib/.../myfile.csv'),
Layouts.ip_claim_layout,
      CSV(HEADING(0)) );

oDSDistributed := DISTRIBUTE(oDS, bene_id);
OUTPUT(oDSDistributed,, '~somewhere::over::here::myfile', OVERWRITE);
```

- ECL to use data loaded into cluster:

```
oDS := DATASET('~somewhere::over::here::myfile', Layouts.ip_claim_layout);
```

ARCHWAY HEALTH ADVISORS

SuperFiles

- Super File = Symbolic link, list of sub-files
- Each sub-file must have the same layout

```
WEBLOGS_FILE := '~somewhere::logs::web'
Std.File.CreateSuperFile(WEBLOGS_FILE);
...

run_report() := FUNCTION
 oDS := DATASET(WEBLOGS_FILE, Layouts.weblogs_layout, CSV);
 RETURN TABLE(oDS, { ip_address; COUNT(GROUP); }, ip_address );
END;
```

- Including (more) data:

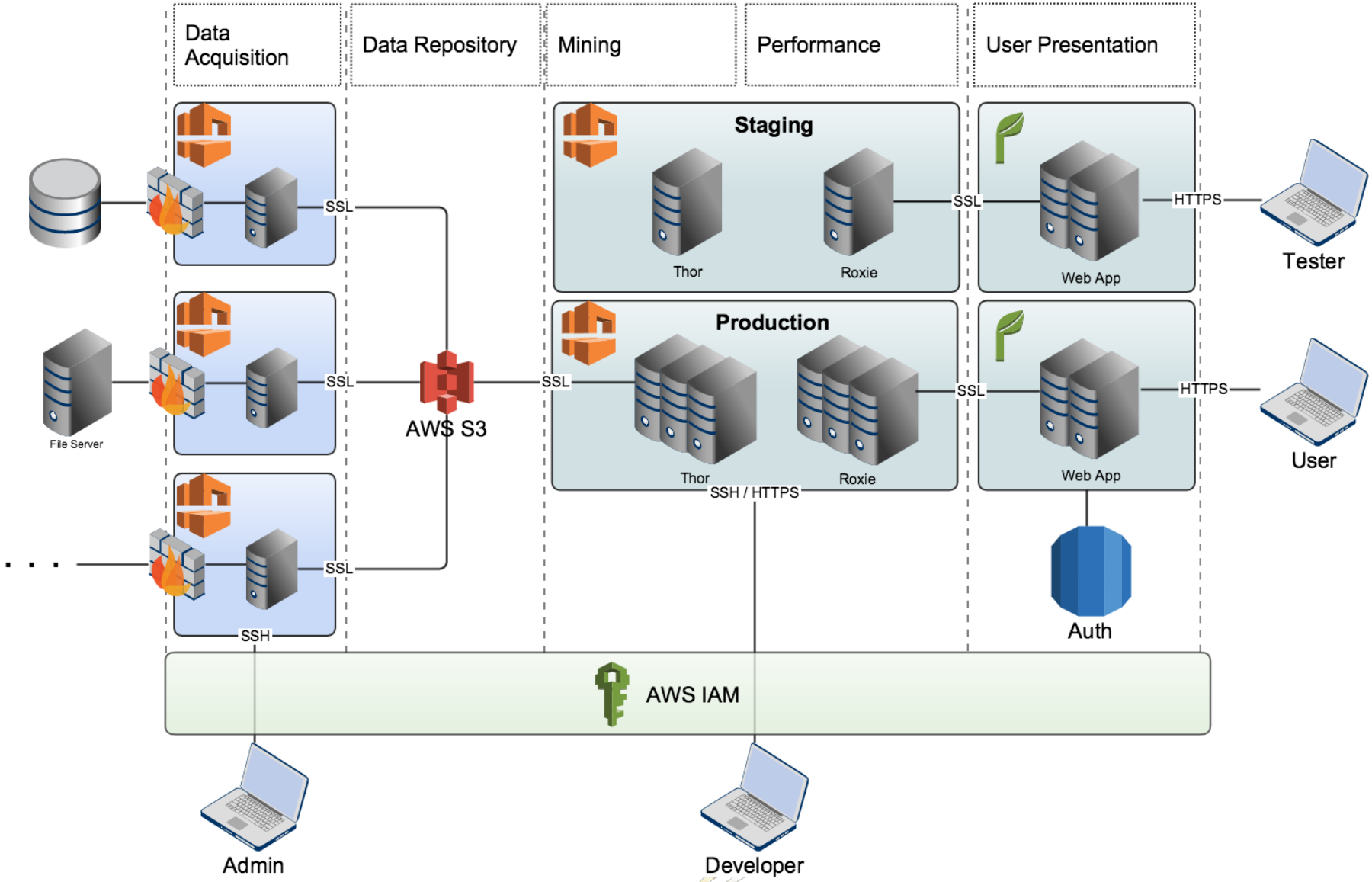
```
SEQUENTIAL(
      Std.File.StartSuperFileTransaction(),
      Std.File.AddSuperFile(WEBLOGS_FILE, '~somewhere::logs::web::2015::04::01'),
      Std.File.FinishSuperFileTransaction()
);
```

Data Services: Reusability

```
EXPORT run_it( pServiceName, pParams,  pReportFunction, pSortByField) := MACRO

      // Filtering data based on parameters
      #UNIQUENAME(DS);
      %DS% := WS.Datasets.dsFactEpisodeCostsIndex;
      [...]
      #UNIQUENAME(B)
      %B% := IF(COUNT(pParams[1].providers) = 0, %A%, %A%(provider_id IN pParams[1].providers));
      #UNIQUENAME(C)
      %C% := IF(COUNT(pParams[1].npis) = 0, %B%, %B%(at_npi IN pParams[1].npis OR op_npi IN pParams[
      [...]
      #UNIQUENAME(report)
      %report% := pReportFunction(%K%);

      #UNIQUENAME(sorted)
      %sorted% := SORT(%report%, pSortByField);
      #UNIQUENAME(O1)
      %O1% := OUTPUT(pParameters, NAMED('Request'));
      oSummary := DATASET([{ COUNT(%sorted%) }], WS.Layouts.service_summary_layout);
      #UNIQUENAME(O2)
      %O2% := OUTPUT(oSummary, NAMED('Metadata'));
      #UNIQUENAME(O3)
      %O3% := OUTPUT(
            CHOOSEN(%sorted%, pParams[1].results, pParams[1].offset),
            NAMED(pServiceName), ALL);
     PARALLEL(%O1%, %O2%, %O3%);
ENDMACRO;
```

ARCHWAY HEALTH ADVISORS

AHA System Architecture



Archway Analytics

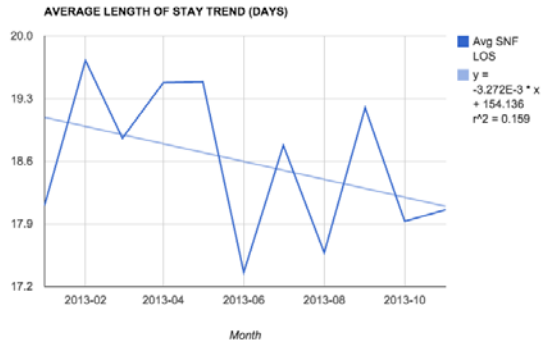
By DRG

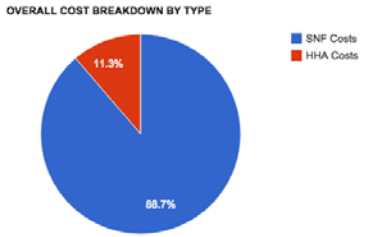
Cost Trends

By Bundle

By DRG

Spending Deep Dive



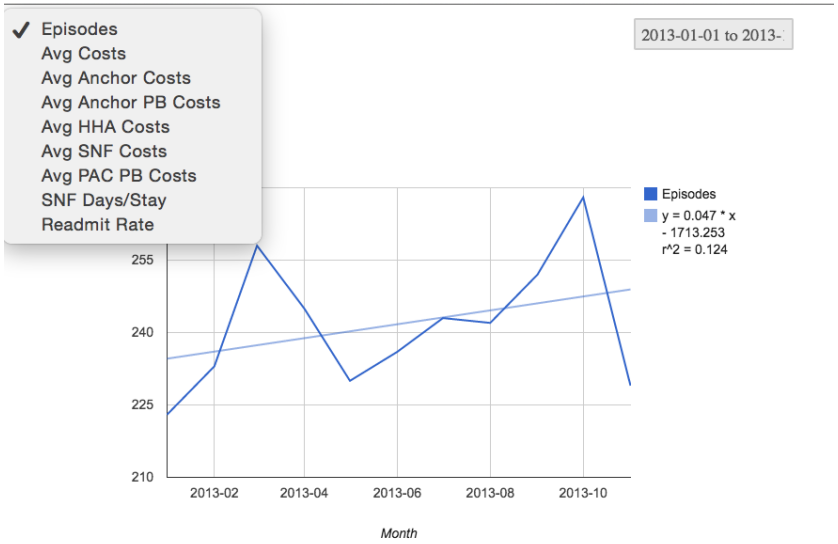


TOTAL PROGRAM SUMMARY

Volume	Total Cost	Avg Cost/Episode	Avg SNF LOS
2,659	$59,073,970	$22,217	18.60

TOP BUNDLES

Bundle	Volume	Total Costs	Avg Costs	Avg SNF LOS
Amputation	178	$4,103,902	$23,056	22.35
Medical non-infectious orthopedic	217	$3,993,311	$18,402	17.50
Other respiratory	151	$3,353,616	$22,209	16.83
Coronary artery bypass graft	101	$2,228,468	$22,064	18.14
Stroke	96	$2,202,995	$22,948	20.58



Bundles

Chart Options

Bundle	Episodes	Avg Costs	Avg Anchor Costs	Avg Anchor PB Costs	Avg SNF Costs	Avg HHA Costs	Avg PAC PB Costs	Avg SNF Day Per Stay
Amputation	178	$23,056	$0	$0	$20,702	$2,353	$0	2
Medical non-infectious orthopedic	217	$18,402	$0	$0	$16,037	$2,365	$0	1
Other respiratory	151	$22,209	$0	$0	$19,454	$2,755	$0	1
Coronary artery bypass graft	101	$22,064	$0	$0	$19,649	$2,415	$0	1
Stroke	96	$22,948	$0	$0	$20,441	$2,507	$0	2
Simple pneumonia and respiratory infections	84	$26,032	$0	$0	$23,478	$2,553	$0	2
Cardiac								

POWERED BY HPCC SYSTEMS® hpccsystems.com

ARCHWAY HEALTH ADVISORS

Thank you

Questions? Feedback?

Questions ? Feedback ?

lpezet@archwayha.com

www.linkedin.com/in/lucpezet

mezzetin.blogspot.com

www.linkedin.com/in/lucpezet

HPCC Systems open source portal:

http://hpccsystems.com

mezzetin.blogspot.com